PRIOR-INFORMED MULTIVARIATE LSTM (PIM-LSTM) FOR ECONOMIC TIME SERIES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

PETEK AYDEMİR AYDIN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
STATISTICS

MARCH 2025

Approval of the thesis:

PRIOR-INFORMED MULTIVARIATE LSTM (PIM-LSTM) FOR ECONOMIC TIME SERIES

submitted by **PETEK AYDEMİR AYDIN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** i**n Statistics, Middle East Technical University** by,

Prof. Dr. Naci Emre Altun
Dean, **Graduate School of Natural and Applied Sciences** _____

Prof. Dr. Vilda Purutçuoğlu
Head of the Department, **Statistics** _____

Prof. Dr. Ceylan Yozgatlıgil
Supervisor, **Statistics, METU** _____

Examining Committee Members:

Prof. Dr. Özlem İlk Dağ
Statistics, METU _____

Prof. Dr. Ceylan Yozgatlıgil
Statistics, METU _____

Prof. Dr. Seher Nur Sülkü
Econometrics, HBV Uni. _____

Assoc. Prof. Dr. Hande Alemdar
Computer Engineering, METU _____

Assoc. Prof. Dr. Kamil Demirberk Ünlü
Industrial Engineering, Atılım Uni. _____

Date: 06.03.2025

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last name : Petek Aydemir Aydın

Signature :

ABSTRACT

PRIOR-INFORMED MULTIVARIATE LSTM (PIM-LSTM) FOR ECONOMIC TIME SERIES

Aydemir Aydın, Petek
Doctor of Philosophy, Statistics
Supervisor: Prof. Dr. Ceylan Yozgatlıgil

March 2025, 103 pages

Deep Learning is a subset of machine learning that emphasizes algorithms influenced by the human brain, called artificial neural networks. Physics-Informed Neural Networks (PINNs) represent a distinct deep learning method that combines the strengths of neural networks with the physical principles that dictate particular systems. The main goal of this thesis is to enhance the PINNs model for multivariate time series by integrating causal relationships and cross-correlations to improve overall model performance. For this purpose, we developed a Prior-Informed Multivariate Long Short-Term Memory (PIM-LSTM) model. First, its application to the New Keynesian and Dividend-Augmented Goodwin-Keen (DAGKM) models is demonstrated. Then, the forecast performance of the PIM-LSTM model is compared to the LSTM and PINN models. Our findings indicate that the PIM-LSTM model demonstrates strong predictive performance on the New Keynesian Model for Turkiye and Mexico's macroeconomic series, achieving lower MAE, RMSE, and MASE compared to LSTM and PINNs models. The PIM-LSTM model also performs well in the DAGKM model. Integrating the New Keynesian model for

Turkiye and Mexico enhances the analysis by capturing country-specific monetary policies and economic dynamics. Similarly, incorporating the DAGKM model enhances the analysis by capturing cyclical growth and income distribution.

Keywords: Time series analysis, Forecasting Methods, Deep Learning Algorithms

ÖZ

ÖN BİLGİYE DAYALI ÇOK DEĞİŞKENLİ LSTM (PIM-LSTM) İLE EKONOMİK ZAMAN SERİLERİ

Aydemir Aydın, Petek
Doktora, İstatistik
Tez Yöneticisi: Prof. Dr. Ceylan Yozgatlıgil

Mart 2025, 103 sayfa

Derin Öğrenme, yapay sinir ağları olarak adlandırılan insan beyninden etkilenen algoritmaları vurgulayan makine öğreniminin bir alt kümesidir. Fizik Bilgilendirmeli Sinir Ağları (PINNs), sinir ağlarının güçlü yönlerini belirli sistemleri belirleyen fiziksel ilkelerle birleştiren farklı bir derin öğrenme yöntemini temsil eder. Bu tezin temel amacı, genel model performansını iyileştirmek için nedensel ilişkileri ve çapraz korelasyonları entegre ederek çok değişkenli zaman serileri için PINNs modelini geliştirmektir. Bu amaçla, Ön Bilgiye Dayalı Çok Değişkenli Uzun Kısa Vadeli Bellek (PIM-LSTM) modeli geliştirilmiştir. İlk olarak, Yeni Keynesyen ve Temettü Artırılmış Goodwin-Keen (DAGKM) modellerine uygulanması gösterilmiştir. Ardından, PIM-LSTM modelinin tahmin performansı LSTM ve PINNs modelleriyle karşılaştırılmıştır. Bulgularımız, PIM-LSTM modelinin Türkiye ve Meksika'nın makroekonomik serileri için Yeni Keynesyen Model üzerinde güçlü tahmin performansı sergilediğini ve LSTM ve PINN modellerine kıyasla daha düşük MAE, RMSE ve MASE elde ettiğini göstermektedir. PIM-LSTM modeli DAGKM modelinde de iyi performans göstermektedir. Türkiye ve Meksika için Yeni

Keynesyen modelin entegre edilmesi, ülkeye özgü para politikalarını ve ekonomik dinamikleri yakalayarak analizi geliştirmektedir. Benzer şekilde, DAGKM modelinin dahil edilmesi, konjonktürel büyüme ve gelir dağılımını yakalayarak analizi geliştirmektedir.

Anahtar Kelimeler: Çok Değişkenli Zaman Serileri Analizi, Öngörü Metotları, Derin Öğrenme Algoritmaları

To My Family

ACKNOWLEDGMENTS

Initially, I would like to sincerely express my endless appreciation to my thesis advisor Prof. Dr. Ceylan Yozgatlıgil for assists, patience and recommendations during my thesis. This thesis would not finish without her encouranging behaviours, invaluable help and guidance. I feel honored to work with her and benefit from her experience and knowledge. She is more than thesis advisor to me with her friendly and sympathetic attitudes.

I would also like to present my appreciative thanks to my examining committee members, Prof. Dr. Özlem İlk Dağ, Assoc. Prof. Dr. Hande Alemdar, Prof. Dr. Seher Nur Sülkü, and Assoc. Prof. Dr. Kamik Demirberk Ünlü for their participation, valuable and constructive feedback, suggestions, and comments.

I am grateful to my friends Neşe Bayram, Serenay Çakar, Rana Arslan, İrem Tanrıverdi and Ozancan Özdemir for their support and motivation during my thesis. Moreover, I would like to present my special gratitude to my instructors for helping me to enhance my knowledge in this field.

I owe special thanks to Eray Aydın for his patience and loving support during this thesis, I want to present my thanks to my friends Duygu Uzunlar Atamtürk and Hüseyin Atamtürk who do not leave me alone during this thesis period.

Finally, I want to express my grateful thanks to my dear mother Gülcan Aydemir and my dear father Yıldırım Aydemir who have provided me through endless love, emotional and unconditional support in my life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

AC-OPF	AC-Optional Power Flow
ACF	Autocorrelation Function
AD	Automatic Differentiation
ADF	Augmented Dickey Fuller
ALDL	Augmented Lagrangian Deep Learning
APINN	Augmented PINN
AR	Autoregressive
ARMA	Autoregressive Moving Average
ARIMA	Autoregressive Integrated Moving Average
ANN	Artificial Neural Networks
BNN	Bayesian Neural Network
B-PINN	Bayesian PINN
CNN	Convolutional Neural Network
cPINN	Conservative PINN
DAGKM	Dividend-Augmented Goodwin-Keen Model
DGM	Deep Galerkin Methods
DL	Deep Learning
DSGE	Dynamic Stochastic General Equilibrium
FFNN	Feed-Forward Neural Network
FORM	First-Order Reliability Method
fPINNs	Fractional PINNs
FRED	Federal Reserve Economic Data
hp-VPINN	High-Order Polynomial Variational PINN
IPINN	Improved PINN
IS	Investment-Savings
LRS	Locally Refined Sampling Strategy

LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MASE	Mean Absolute Scaled Error
MLP	Multi-Layer Perceptron
MO-PINNs	Multi-Output PINNs
MO-MPINNs	Multi-Output Multi-Physics-Informed Neural Networks
MSE	Mean Squared Error
ND	Numerical Differentiation
NKPC	New Keynesian Philips Curve
NSE	Navier- Stokes Equations
PACF	Partial Autocorrelation Function
PDEs	Partial Differential Equations
PIM-LSTM	Prior-Informed LSTM
PINNs	Physics Informed Neural Networks
PINNs-WE	PINNs with Equation Weighting
QML	Quantum Machine Learning
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SA-PINNs	Self-Adaptive Weights PINNs
TSF	Time Series Forecasting
WSS	Wall Shear Stress
XPINNs	Extended PINNs
VAR	Vector Autoregressive
VECM	Vector Error Correction Model

CHAPTER 1

INTRODUCTION

Since the dawn of humanity, people have wondered about the future and tried to predict what will happen. Forecasting has attracted the interest of researchers and experts in many fields and has become an important activity in economics, business, marketing and various disciplines. This study focuses on forecasting through time series analysis. A time series consists of data points gathered at consecutive time intervals, which may be equally or unequally spaced. Time series analysis is a statistical method used to examine a sequence of data points collected over time, aiming to identify patterns, trends, and other characteristics within the data. It plays a significant role in understanding the temporal dynamics of the dataset, which can then be used to make informed decisions. The primary goal of time series analysis is to obtain reasonably accurate forecasts. Time series forecasting entails constructing models based on historical data and using these models to predict future observations. Time series analysis can be categorized into univariate and multivariate analyses. Univariate time series analysis deals with a time-dependent variable and focuses on understanding its behavior and making predictions based solely on its past values. The traditional univariate models, including Autoregressive (AR), Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA) introduced by Box and Jenkins [5], and Exponential Smoothing introduced by Brown [6], are widely opted for univariate time series forecasting (TSF). Multivariate time series analysis involves multiple interrelated time-dependent variables, where the relationships and dependencies between these variables are also considered. This added complexity requires more sophisticated modeling techniques but can provide more accurate and comprehensive forecasts by leveraging the interactions between the variables. The vector autoregressive (VAR) model proposed by Sims [39] is mostly preferred traditional model for multivariate

time series forecasting. It is the multivariate version of the AR model. In a VAR model, each variable is represented as a linear function of its own past values and the past values of all other variables.

VAR models are particularly effective in describing the dynamic behavior of time series and producing better forecasts compared to univariate models. However, the limitation of this model is that it only works well with stationary series. The second classical multivariate time series forecasting method is the Vector Error Correction Model (VECM). It is a cointegrated VAR model. VECM is an advanced statistical technique designed to analyze and predict multivariate time series data that exhibit a cointegration. Cointegration indicates a long-run association among variables, despite short term fluctuations. The VECM is particularly useful in economic and financial contexts, where variables often have long-run interrelationships, such as interest rates, exchange rates, and prices. Despite VECM offering a comprehensive and accurate framework for understanding and forecasting complex time series by grasping not only short-term variations but also long-term trends, it has certain limitations: it is only effective with difference-stationary series.

Traditional methods for multivariate time series analysis have been limited since real-world time series have undesirable characteristics such as non-stationarity, seasonality, irregular fluctuations and cyclical variations. These methods fail to capture these undesirable characteristics and need large datasets to accurately capture the interrelationship between the variables. They are also sensitive to outliers. Due to these limitations, deep learning methods have become the preferred solution. Deep learning (DL), a subfield of machine learning, has revolutionized data analysis and prediction capability. Deep learning utilizes multilayer artificial neural networks to develop algorithms that extract meaningful patterns and features from large data sets. It is effectively used in various areas, including natural language processing (NLP), image processing, speech recognition, gaming, and autonomous vehicles.

The main strength of this field is its capacity to handle intricate, high-dimensional data. Unlike traditional machine learning algorithms, deep learning models can

automatically extract features from data. This reduces the need for manual feature engineering and enhances the model's overall performance. Deep learning models employ different types of artificial neural networks, the most widely common of which are Multi-Layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs), and Transformers. The success of deep learning is directly tied to the presence of large datasets and powerful computational resources. Large datasets enable models to make more general and accurate predictions, while powerful computational resources speed up the training process of these models. For this reason, deep learning is a constantly evolving and expanding field in machine learning. As new algorithms and architectures are developed, the application areas of deep learning are expanding and providing effective solutions in more industries. As part of these advancements, Physics-Informed Neural Networks (PINNs) utilize DL techniques to develop more precise and interpretable models of physical systems. PINNs incorporate physical laws and differential equations directly into the training process of neural networks, ensuring that predictions align with established scientific principles while learning from data. By incorporating known deterministic relationships between variables into neural networks, PINNs leverage this additional information to typically enhance the prediction and forecasting performance of the models. PINNs not only learn from data but also make predictions that align with physical laws, providing more profound solutions to scientific and engineering challenges. This approach also allows PINNs to be applied in complex economic models, facilitating more accurate and theoretically consistent modeling of economic dynamics.

Despite the importance of multivariate time series forecasting, the literature lacks sufficient models that effectively capture these complex relationships. Most existing models either ignore or oversimplify these interactions, creating a significant opportunity for advancements in forecasting methodologies that better incorporate variable interdependencies.

The primary objective of this thesis is to propose a novel Multivariate Physics-Informed Neural Network (PINN) model for multivariate time series forecasting, considering the interdependencies and cross-correlations among variables. The proposed model aims to enhance forecasting accuracy by explicitly capturing the relationships among multiple variables. Additionally, this model will be applied to the New Keynesian model and the Dividend-Augmented Goodwin-Keen model, which have not yet been explored with such approaches in the context of economic data. The applications of this method to economic data aim to provide new insights into forecasting within this theoretical framework.

The New Keynesian macroeconomic model is a development of Keynesian economics, merging microeconomic theory with macroeconomic consequences. It underscores the significance of market imperfections, specifically price and wage rigidities, in explaining why the economy may not always achieve full employment, and how monetary policy can impact output and inflation. It provides a framework for analyzing monetary and fiscal policy, illustrating how these policies can stabilize the economy by affecting aggregate supply and demand. The fundamental assumption of New Keynesian macroeconomic analysis is the rational expectations of households and firms. The New Keynesian model includes three core equations: the New Keynesian Phillips Curve (NKPC), the dynamic Investment-Savings (IS) equation, and Taylor's Rule. Together, these equations depict the behavior of inflation, output, and interest rates, offering a comprehensive perspective of economic dynamics. The NKPC explains that the expected inflation one period ahead and the output gap influence the current inflation. The dynamic IS equation states that the current output gap is a function of the expected output gap one period ahead, expected inflation, the nominal interest rate, and the natural rate of interest. The difference between the output and the potential output is called the output gap. The Taylor rule explains how nominal interest rates are determined. According to the Taylor rule, nominal interest rates depend on the current inflation rate and the output gap.

The Dividend-Augmented Goodwin-Keen Model (DAGKM) extends the traditional Goodwin-Keen model by incorporating dividend payments into the analysis of economic cycles, particularly focusing on the interactions between wage share, employment rate, and debt ratio. This enhancement acknowledges that firms invest their profits and distribute a portion as dividends to shareholders, influencing debt accumulation dynamics. Key components of the DAGKM model are wage share, employment rate and debt ratio. The wage share is a fraction of the total output allocated to employee compensation. The employment rate is the ratio of employed individuals to the total labor force. The debt ratio is the proportion of corporate debt relative to total economic output. By integrating dividend payments into the Goodwin-Keen framework, the DAGKM Model offers a more comprehensive understanding of the interplay between corporate financial policies and macroeconomic cycles. This enhancement underscores the importance of considering dividend distributions when analyzing the sustainability and stability of economic growth patterns.

This study seeks to leverage Physics-Informed Neural Networks (PINNs) for analyzing time series data, aiming to enhance the predictive accuracy and theoretical consistency of both the New Keynesian and Dividend-Augmented Goodwin-Keen models. In this framework, we propose the Prior-Informed Multivariate LSTM (PIM-LSTM), a novel deep learning architecture designed to incorporate prior knowledge constraints into multivariate time-series forecasting. The proposed PIM-LSTM model, inspired by Physics-Informed Neural Networks (PINNs), leverages prior knowledge constraints and incorporates cross-correlation function (CCF) constraints to improve multivariate time-series forecasting.

The remaining part of this thesis is organized as follows: Chapter 2 provides a review of the literature on PINNs models. Chapter 3 discusses the New Keynesian model, the DAGKM model, DL methods, including LSTM and PINNs models and architecture of PIM-LSTM model. Chapter 4 provides an empirical analysis utilizing the PIM-LSTM model to forecast Turkey's output, inflation, and nominal interest rate within the context of the New Keynesian model. It also includes an empirical

analysis using the same PIM-LSTM model to forecast Mexico's output, inflation, and nominal interest rate under the New Keynesian framework. The PIM-LSTM model is also applied to forecast the debt ratio, employment rate and wage share in the United States for the DAGKM model. It also includes a comparison of the out-of-sample performance of all models. Finally, Chapter 5 includes the conclusion and further discussion.

CHAPTER 2

LITERATURE REVIEW

In scientific computing, deep neural networks have been explored for solving partial differential equations (PDEs) through various methods, including augmented Lagrangian deep learning (ALDL) [16], Neural Operators [24, 26], deep Galerkin methods (DGM) [40], and Physics-Informed Neural Networks (PINNs) [48]. Among these, PINNs have emerged as a leading approach due to their ability to seamlessly integrate physical laws into the learning process, making them highly effective for solving complex PDEs.

PINNs offer several advantages, including robust generalization, automatic differentiation, and grid-free capabilities [45, 34]. By incorporating physical laws directly into the neural network framework, PINNs use a loss function that combines residuals from differential equations and boundary conditions, eliminating the need for simulation or experimental data. This approach reduces reliance on large labeled datasets, enhances computational efficiency, and ensures physically consistent solutions. Additionally, PINNs' ability to generalize well to unseen data makes them more versatile. Their grid-free nature allows them to be applied to complex geometries, and the use of automatic differentiation simplifies the computation of derivatives. These features make PINNs highly suitable for a wide range of scientific applications, particularly in data-scarce environments.

PINNs have been the subject of increasing amounts of research in many different fields, especially in engineering and computational sciences. In their review article, Lawal et al. (2022) [25] state that 288 documents were chosen from the literature on these disciplines. Their review article offers a perceptive depiction of the development of PINNs over the previous three and a half years, showing a consistent

increase in publications from 2019 to mid-2022, with a peak in 2021. This pattern demonstrates how scholars are becoming more interested in and dedicated to using PINNs to solve challenging issues and improve their work. In this part, the most recent studies on PINNs from different research areas as well as studies including its hybrid forms will be reviewed.

2.1 Related Studies

In their paper, Raissi et al. (2019) [35] introduced the concept of PINNs to enrich deep learning. To this end, they employed deep neural networks to address supervised learning tasks by incorporating established principles from mathematical physics, specifically those defined by general nonlinear partial differential equations (PDEs). Based on the characteristics and structure of their available data, they developed two distinct types of algorithms: one for continuous time models and another for discrete-time models. They emphasize that their proposed method should not be viewed as a replacement for traditional numerical techniques used to solve PDEs, such as finite element or spectral methods. Rather, they demonstrate that classical methods, including Runge-Kutta time-stepping schemes, can effectively complement deep neural networks. Their integration provides valuable insights for creating structured predictive algorithms. Additionally, the simplicity of implementing neural networks supports the rapid development and testing of new ideas, potentially initializing a new era of data-driven scientific computing.

In their paper, Mao et al. (2020) [28] investigate the use of PINNs to the solution of forward and inverse Euler equation problems for high-velocity aerodynamic flows. Despite discontinuities such as oblique shock waves, they show that PINNs can solve the forward issue and capture solutions with little scattered input. Motivated by Schlieren photography, PINNs for the inverse issue perform better than conventional techniques in situations where standard techniques are inadequate. They do this by reliably inferring density, velocity, and pressure fields from data on density gradients and pressure. While their study also demonstrates that employing Euler equations in

characteristic form outperforms the conservative form, PINNs still lag behind conventional approaches for forward issues. This suggests that PINNs may be useful in the future for aerodynamic analysis applications, as they are adept at resolving challenging inverse problems.

Despite notable advancements in simulating flow problems through the numerical discretization of the Navier–Stokes equations (NSE), Cai et al. (2021) [7] discuss the existing challenges in their paper. For instance, they stated that current methods still struggle with integrating noisy data, managing the complexity of mesh generation, and addressing high-dimensional problems governed by parametrized NSE. To overcome these issues, the authors proposed a method called flow physics-informed learning, which utilizes PINNs to integrate data and mathematical models effectively. Their findings indicate that this approach successfully infers hidden velocity fields and unknown parameters of PDEs from sampled data alone, indicating the potential of PINNs to address the limitations of conventional numerical simulations.

Cai et al. (2021) [8] applied the PINN framework to address several representative heat transfer challenges. They utilized PINNs as a transformative solution for bridging the gap between experimental and computational heat transfer. By integrating conservation laws directly into their architecture and utilizing sparse measurements through multifidelity methods, PINNs enable accurate inference of velocity and temperature fields, as well as unknown thermal boundary conditions or interfaces. Their effectiveness is particularly evident in handling realistic conditions that challenge conventional computational methods. Applications in power electronics highlight the practical utility of PINNs in addressing complex heat transfer problems encountered in industrial settings. The collective findings confirm that PINNs not only excel in solving ill-posed problems difficult for traditional methods but also effectively bridge the gap between computational and experimental heat transfer.

Tanios (2021) [41] explores the application of PINNs for pricing multi-asset European options within the high-dimensional Black-Scholes and Heston models. Given the recent success of PINNs in approximating solutions to PDEs, their utility extends to accurately estimating solutions and unknown model parameters from observed data. They demonstrate how PINNs offer a straightforward method for calculating the Greeks—quantities that measure the sensitivity of a derivative's price, represented as partial derivatives (either first-order or higher) in calculus.

To improve the quantification of near-wall blood flow and wall shear stress (WSS), which are difficult to quantify yet essential for understanding cardiovascular disorders, Arzani et al. (2021) [1] utilize PINNs. Conventional approaches to patient-specific computational and experimental WSS assessment suffer from limited resolution, noise, and uncertainty. By combining mathematical equations—more specifically, the Navier-Stokes equations that govern blood flow—with sparse measurement data, PINNs provide a strong deep learning methodology. Their study shows that by absorbing a few measurement points, PINNs can effectively handle blood flow issues when inlet and output boundary conditions are unknown. This is especially useful because patient-specific fluid dynamics models sometimes have unknown boundary conditions. Their examples from idealized models of stenosis and aneurysms show how limited measurements combined with a partial understanding of flow mechanics can produce accurate near-wall blood flow data. This hybrid data-driven and physics-based approach has significant potential to advance high-fidelity modeling of near-wall hemodynamics in cardiovascular disease.

Jiang et al. (2022) [20] investigate nonlinear dynamics inside optical fibers while utilizing PINNs to solve the nonlinear Schrödinger equation. The paper offers a thorough assessment of PINNs' abilities to handle a range of physical phenomena, including higher-order nonlinear effects, dispersion, and self-phase modulation. The study looks at both soliton and multipulse propagation, and it finds that adding physical characteristics to the PINN as extra input controls, such as pulse peak power and subpulse amplitudes, greatly improves the network's generalization capabilities

in many contexts. The shortcomings of earlier models—which were frequently limited to isolated instances—are addressed by this approach. The findings show that, in comparison to the split-step Fourier method, PINNs require significantly less data and have lower computational complexity, making them an effective tool for solving PDEs and advancing the fields of scientific computing and automated modeling in fiber optics.

To accurately solve the AC-Optimal Power Flow (AC-OPF) problem which is often challenging due to its non-linear and non-convex structure, Nellikkath and Chatzivasileiadis (2022) [31] trained PINNs. By embedding these equations directly into the training process, the dependency on the size of the training data set is substantially reduced, making the model more efficient and practical for real-world applications. Moreover, this integration significantly improves the worst-case performance of neural networks, ensuring more reliable and robust outcomes in power system operations. This development represents a crucial step forward in enhancing the effectiveness and dependability of neural networks in managing complex power systems.

Bararnia and Esmaeilpour (2022) [4] investigate the use of PINNs to solve challenging thermal-fluid issues, with a particular emphasis on thermal and viscous boundary layers. The three benchmark problems they chose to study the impact of unbounded boundary conditions and equation nonlinearity on the width and depth of the network structure were Blasius-Pohlhausen, Falkner-Skan, and Natural Convection. They used big-data training using TensorFlow to create and train the PINN models, revealing hidden physics in transport phenomena. The correctness and dependability of the PINN models were confirmed by comparing their predictions with the outcomes of using Richardson extrapolation in conjunction with the finite difference technique. Key findings showed that the number of neurons and layers needed in a neural network to produce precise solutions is highly influenced by the Prandtl number in the heat equation. Furthermore, to effectively manage the infinite boundary condition, handling unbounded boundary conditions by placing the boundary farther from the origin required adding more layers and neurons. The PINN

models that had been trained were effectively utilized for assessing boundary layer thicknesses on previously unseen data, proving their resilience and usefulness. In addition to providing insights into how neural network topologies can be optimized for best performance, this research shows how PINNs can be used to solve extremely nonlinear and complex boundary layer issues in thermal-fluid dynamics.

Jeong et al. (2024) [19] applied PINNs to overcome a complex multi-physics problem involving electromagnetism, fluid dynamics, and heat transfer. They applied this model to a cylindrical conductor, considering the interplay between electrical and magnetic fields and the thermal interactions between the conductor and its environment. To enhance the performance of their PINN, the authors divided the problem into seven interlinked neural networks. They employed domain decomposition and variable separation techniques, optimizing each network individually and ensuring efficient data transfer between them. Their results demonstrate impressive accuracy, with less than 2% error compared to traditional numerical methods and analytical solutions.

Recent advancements have introduced extensions to the traditional PINN framework, aiming to enhance its capabilities further. In addition to papers utilizing traditional PINNs in different research fields, we also review studies focused on developing hybrid forms of PINNs to improve their predictive capability under different circumstances (see e.g. [32], [2], [46], [23]). For instance, Pang et al. [32] extend PINNs to fractional PINNs (fPINNs) which will be efficient in solving space-time fractional advection-diffusion equations (fractional ADEs). They demonstrate the accuracy and effectiveness of fPINNs in solving multidimensional forward and inverse problems with forcing terms whose values are only known at randomly scattered spatio-temporal coordinates. Additionally, they introduce a hybrid approach that involves constructing the residual in the loss function using both automatic differentiation for the integer-order operators and numerical discretization for the fractional operators. Their proposed PINNs solve several inverse problems in one, two, and three dimensions, enabling the identification of the fractional orders,

diffusion coefficients, and transport velocities while obtaining accurate results under proper initializations even in the presence of significant noise.

Bai et al. (2022) [2] propose an improved PINN (IPINN), which integrates a local adaptive activation function for neurons to enhance neural network performance. They successfully applied the IPINN to the Ivancevic option pricing model and the Black–Scholes model in finance, demonstrating its effectiveness in improving accuracy and efficiency for complex financial problems. They address the rogue wave and soliton solutions of the Ivancevic option pricing model, as well as the numerical solution of the Black–Scholes model using IPINN method. Their numerical experiments demonstrate that the IPINN method exhibits faster convergence, greater stability, and higher accuracy compared to the traditional PINN method.

Yang et al. (2021) [46] introduce a Bayesian PINN (B-PINN) for addressing both forward and inverse nonlinear problems characterized by PDEs and noisy data. In their Bayesian framework, the prior is established by combining a Bayesian neural network (BNN) with a PINN for PDEs, while the posterior can be estimated using either Hamiltonian Monte Carlo or variational inference. Compared to PINNs, their results indicate that B-PINNs not only provide better uncertainty quantification but also deliver more accurate predictions in high-noise scenarios, thanks to their ability to mitigate overfitting.

Kharazmi et al. (2021) [23] develop a high-order polynomial Variational PINN (hp-VPINN) method integrating a variational formulation based on the sub-domain Petrov–Galerkin method, where neural networks are used as the trial space and localized non-overlapping high-order polynomials form the test space. This innovative combination enhances the method's ability to model PDE solutions accurately by utilizing the complementary strengths of neural networks and polynomial functions. The use of integration-by-parts, which increases performance in processing difficult or rough solutions, such as singularities, steep gradients, and abrupt changes in the data, is a fundamental component of hp-VPINN. Their

comparative analysis with traditional PINNs highlights the hp-VPINN's improved accuracy and efficiency, confirming its robustness in practical applications.

Jagtap et al. (2020) [18] introduce an innovative method called the conservative PINN (cPINN), designed to handle nonlinear conservation laws in discrete domains. In this approach, discrete domains refer to the segmented regions of the computational domain created by partitioning. The cPINN enforces conservation principles by maintaining flux continuity in the strong form at the interfaces between these segments. Furthermore, the cPINN utilizes locally adaptive activation functions, which accelerates the training process compared to traditional fixed activation functions.

In 2020, Jagtap and Karniadakis [17] propose the extended PINNs (XPINNs) framework, a novel development that generalizes and improves upon existing PINN and cPINN methodologies by incorporating a more flexible and comprehensive space-time domain decomposition approach. XPINNs leverage multiple neural networks deployed in smaller subdomains, enhancing representation capacity and parallelization. This approach reduces training costs and improves computational efficiency by allowing parallel processing across both spatial and temporal domains.

Considering the challenges of PINNs and XPINNs in achieving optimal performance, particularly in handling complex domain decompositions and parameter sharing, Hu et al. (2023) [15] introduce the augmented PINN (APINN), a novel approach designed to enhance their capabilities. One of the key innovations of APINN is its ability to utilize all available training data across the entire domain, rather than limiting data usage to specific subdomains. This approach enhances the efficiency of the learning process and improves the generalization of the model. Additionally, APINN employs parameter sharing across sub-networks to capture common features and components in decomposed functions, thus boosting the overall performance and generalization capability. Their approach addresses key limitations of existing methods and opens new places for applying PINNs to complex scientific and engineering problems.

Wandel et al. (2022) [44] introduce an innovative approach by combining PINNs and convolutional neural networks to deal with the challenge of solving PDEs. The approach produces rapid and continuous solutions which can be applicable across diverse domains. They illustrate their methodology by demonstrating the incompressible Navier-Stokes equation and the damped wave equation. By utilizing spline-PINNS, their model can capture various phenomena, reduce the accuracy gap in computational fluid Dynamics and also work faster.

In their study, Chiu et al. (2022) [9] suggest CAN-PINN, a novel PINN technique that combines numerical differentiation (ND) with automated differentiation (AD) to improve training accuracy and efficiency. By using differential equations to restrict the training loss function, PINNs make sure their outputs adhere to the laws of physics. Even though AD calculates precise gradients at any point, achieving high accuracy often requires a large number of collocation points; otherwise, it may lead to unphysical solutions when fewer points are used. In order to overcome this, CAN-PINN combines AD and ND, making use of ND's capacity to connect nearby collocation sites for effective training in sparse sample regimes. When compared to ND-based PINNs alone, this hybrid technique produces training that is up to two orders of magnitude more robust and precise. Fluid dynamics challenges such as flow mixing, lid-driven flow, and channel flow over a backward-facing step were investigated with the CAN-PINN framework, which proved to be more accurate and resilient than traditional AD-based PINNs.

Zhang et al. (2022) [49] present GW-PINN, a deep learning technique that does not require labeled data to estimate groundwater flow. GW-PINN employs PINN and modifies its loss function to include either soft or hard constraints. To maximize sampling and training efficiency, it uses a snowball-style two-stage training approach and a locally refined sampling strategy (LRS). According to their results, GW-PINN captures variations in hydraulic heads in a variety of aquifers effectively; the hard constraint outperforms the soft constraint. With the help of the LRS approach and two-stage training, they demonstrate that GW-PINN becomes a more accurate and effective instrument for simulating groundwater flow.

In their paper, McClenny and Braga-Neto (2023) [29] present a unique method of optimizing PINNs using self-adaptive weights (SA-PINNs). By training adaptive weights that are applied to each training point separately, the technique enables the neural network to concentrate on difficult areas of stiff PDEs. By growing when losses are greater, the weights function as a soft attention mask, helping the network to become more accurate in challenging situations. In order to better understand how these weights affect training dynamics, the research also offers a continuous map of these weights using Gaussian Process regression and produces the Neural Tangent Kernel matrix for SA-PINNs. When compared to state-of-the-art PINN algorithms, numerical experiments show that SA-PINNs achieve lower L2 error and require fewer training epochs.

In the same year, Vadyala and Betgeri [43] propose a hybrid Quantum Machine Learning (QML) model named quantum-based PINNs, which combines classical information processing with quantum manipulation and processing, along with PINNs. This model is designed to address challenges related to reliability, trustworthiness, safety, and security in QML while leveraging the strengths of both classical and quantum computing. They achieved the highest performance with their quantum simulation data containing outliers by utilizing a neural network architecture consisting of 6 layers and 40 neurons.

Meng et al. (2023) [30] introduce PINN-FORM, a novel combination of the first-order reliability method (FORM) and PINN designed to tackle the problems associated with structural reliability analysis, especially when working with complex limit state functions expressed as implicit PDEs. These kinds of problems are generally quite computationally challenging using traditional FORM approaches. By using PINN's capabilities as a black-box solution tool, they suggested PINN-FORM to address these problems by doing away with the necessity to compute actual structural answers directly.

In 2024, Liu et al. [27] address the limitation of traditional PINNs in handling discontinuities, particularly when compared to conventional shock-capturing

methods. To overcome this challenge, they develop an innovative approach called PINNs with Equation Weighting (PINNs-WE). This method enhances the capability of PINNs to accurately capture shocks by incorporating equation weighting techniques, which adjust the relative importance of different components in the loss function. By employing this approach, PINNs-WE can more effectively manage abrupt changes and discontinuities in various physical phenomena, significantly enhancing performance where traditional PINNs struggle. They concluded that PINNs-WE provides a more robust and efficient tool for solving problems involving discontinuities.

There are also studies that focus on multi-output approaches within the context of PINNs [47, 13]. In 2022, Yang and Foster [47] proposed the Multi-Output PINNs (MO-PINNs) to solve both forward and inverse PDE problems with noisy data. By utilizing the bootstrap method, their framework translates uncertainty from noisy data into multiple measurements based on prior noise distribution. The network outputs are designed to satisfy both the noisy measurements and the underlying physical laws. Numerical experiments showed that MO-PINNs provided accurate predictions and uncertainty distributions, comparable to traditional methods like finite element methods and Monte Carlo simulations. Their work demonstrates the potential of MO-PINNs for uncertainty quantification and accelerating predictions in engineering applications.

In another study, Hao et al. (2024) [13] introduced the Multi-Output Multi-Physics-Informed Neural Networks (MO-MPINNs) to address challenges in solving the Dimension-Reduced Probability Density Evolution Equation in stochastic dynamical systems. Traditional methods for estimating the intrinsic drift and diffusion coefficients rely on numerical differentiation, which can be unstable and inaccurate, especially in data-scarce regions. MO-MPINNs overcome these issues by integrating multiple outputs within parallel subnetworks, allowing for simultaneous prediction of time-varying coefficients and response probability density functions. This approach embeds physical laws in the loss function and leverages automatic differentiation, providing a more efficient and accurate solution for high-

dimensional, nonlinear systems with complex spatio-temporal dependencies. Their framework enhances the applicability of PINNs to complex stochastic systems with double randomness in parameters and excitations.

Building on these advancements, our study further refines the PINN framework by incorporating multivariate dependencies, allowing for a more comprehensive representation of complex dynamical systems. By embedding cross-correlation relationships within the loss function, we enhance the network's ability to capture intricate interdependencies across multiple outputs. This refinement broadens the applicability of PINNs, extending their use beyond traditional scientific and engineering fields to economic and multi-agent systems, where conventional approaches often fail to address spatio-temporal complexities effectively.

The literature indicates that PINNs are primarily employed for solving engineering and physical problems, including stochastic differential equations, with no known applications in economic modeling. Despite their increasing use in engineering contexts, their potential in economic modeling remains largely unexplored, presenting an opportunity to extend the PINN framework beyond traditional applications. This gap presents an opportunity to extend the PINN framework beyond traditional engineering applications and apply it to economic systems with complex spatio-temporal dynamics in their governing equations.

CHAPTER 3

METHODOLOGY

This chapter provides a comprehensive examination and introduction to the new Keynesian macroeconomic model and the Dividend-Debt-Augmented Goodwin Model (DAGKM). Furthermore, it will delve into the architectures of Long Short-Term Memory (LSTM) networks, Physics-Informed Neural Networks (PINNs) and the proposed model which is called Prior-Informed Multivariate LSTM (PIM-LSTM).

3.1 New Keynesian Macroeconomic Model

The New Keynesian economics is a fusion of Keynesian economics and microeconomic theory. It highlights the importance of market imperfections, particularly inflexible prices and wages, in explaining why the economy may only sometimes achieve full employment, and how monetary policy can impact output and inflation. This framework helps us understand how monetary and fiscal policies can stabilize the economy by affecting overall supply and demand. The New Keynesian view assumes that households and businesses make decisions based on rational expectations. The New Keynesian model comprises three fundamental equations: the New Keynesian Phillips Curve (NKPC), the dynamic Investment-Savings (IS) equation, and Taylor's Rule. Together, these equations explain inflation, output, and interest rates, giving us a clear picture of how the economy works.

The NKPC, originally derived by Roberts in 1995 [36], has found practical application in the New Keynesian Dynamic Stochastic General Equilibrium (DSGE) models. This curve, which states that inflation is influenced by the current output gap and the expectations of the next period's inflation, is derived from the dynamic

Calvo pricing model. Its practical application in economic models is expressed in mathematical terms:

$$\pi_t = \beta E_t\{\pi_{t+1}\} + \kappa\tilde{y}_t + \varepsilon_t^s \quad (1)$$

where $\kappa \equiv \lambda\left(\sigma + \frac{\varphi+\alpha}{1-\alpha}\right)$, $\lambda \equiv \frac{(1-\theta)(1-\beta\theta)}{\theta}\Theta$, $\Theta = \frac{1-\alpha}{1-\alpha+\alpha\epsilon}$

In this equation, β represents discount factor and π_t denotes the inflation, $E_t\{\pi_{t+1}\}$ is the expectation of the next period's inflation, $\tilde{y}_t = y_t - y_t^n$ denotes output gap, y_t denotes output, y_t^n denotes potential output and ε_t^s is supply shock. Additionally, α is capital share, ϵ is elasticity of substitution, φ is Frisch elasticity, σ is relative risk aversion and θ is Calvo parameter. The NKPC equation can also be expressed in terms of output and potential output:

$$\pi_t = \beta E_t\{\pi_{t+1}\} + \kappa y_t - \kappa y_t^n + \varepsilon_t^s \quad (2)$$

The IS equation in economics is an important formulation that helps us to understand how the economy works. It shows that the current output gap is the difference between what we expect from the output gap to be in the next period and a value connected to the difference between the real interest rate $(i_t - E_t\{\pi_{t+1}\})$ and the natural rate of interest. This equation provides a clear understanding of the output gap. This relationship can be expressed mathematically as follows:

$$\tilde{y}_t = -\frac{1}{\sigma}(i_t - E_t(\pi_{t+1}) - r_t^n) + E_t(\tilde{y}_{t+1}) + \varepsilon_t^d \quad (3)$$

where $\tilde{y}_t$ is output gap, $E_t\{\tilde{y}_{t+1}\}$ is expected output gap, i_t is the nominal interest rate, π_t is inflation, r_t^n is the natural rate of interest and ε_t^d is demand shock. The natural rate of interest reflects equilibrium values with flexible prices. The natural rate of interest is calculated by:

$$r_t^n = \rho + \sigma\,\psi_{ya}^n E_t(\Delta a_{t+1}) \quad (4)$$

where $\psi_{ya}^{n} \equiv \frac{1+\varphi}{\sigma(1-\alpha)+\varphi+\alpha}$, $\tilde{y}_t = y_t - y_t^n$, $y_t^n = \psi_{ya} a_t + \psi_y$.

In the equation, the variable a_t represents the total factor productivity shock, which can be described as follows: $a_t = \rho_a a_{t-1} + \xi_t^a$ with $\xi_t^a \sim iid\ N(0, \sigma_a^2)$. ρ_a represents the persistency parameter of total factor productivity shock, and Δ signifies the difference operator.

After applying some calculation, the IS equation can be written in terms of the output:

$$y_t = E_t(y_{t+1}) - \frac{1}{\sigma}(i_t - E_t(\pi_{t+1}) - \rho) + \varepsilon_t^d \tag{5}$$

where $\rho = -\log(\beta)$.

The third equation in the New Keynesian model represents an interest rate rule that explains how the nominal interest rate is established. This rule is commonly associated with the implementation of monetary policy. A commonly used interest rate rule in the literature to represent monetary policy in advanced economies is a Taylor-type rule proposed by Taylor [42]. According to this rule, nominal interest rates increase or decrease in line with the current inflation rate and output gap.

$$i_t = \phi_\pi \pi_t + \phi_y \tilde{y}_t + \varepsilon_t^m \tag{6}$$

Where i_t is the nominal interest rate, π_t is inflation, $\tilde{y}_t$ denotes output gap and ε_t^m is an exogeneous monetary policy shifter. Moreover, ϕ_π is the feedback parameter of inflation and ϕ_y is the feedback parameter of output gap. The interest rate is also re-expressed in terms of the output:

$$i_t = \phi_\pi \pi_t + \phi_y y_t - \phi_y y_t^n + \varepsilon_t^m \tag{7}$$

3.2 Dividend-Augmented Goodwin Keen Model

The Goodwin model proposed by Goodwin [11] is a traditional macroeconomic framework that represents cyclical growth and income distribution by considering capital and labor as interconnected variables. It demonstrates how wages and employment vary over time through a predator-prey dynamic, which illustrates the natural interplay of these economic forces. Building on this foundation, the Goodwin-Keen model proposed by Keen [22] extends the original framework by incorporating private debt. This enhancement allows the model to better capture real-world phenomena, as it shows how debt-financed investment and consumption can either amplify or moderate economic fluctuations. Furthermore, Bailly et al. [3] identified that the original Goodwin-Keen model led to inaccurate estimates because it failed to account for situations in which firms frequently borrow to finance dividends. To address this shortcoming, they proposed a Dividend-Augmented Goodwin model (DAGKM). By incorporating dividends as a share of profits, this revised model produces debt trajectories that more closely align with observed economic behavior.

The dynamic system of DAGKM consists of three equations, which are the wage share (denoted by ω_t), the employment rate (denoted by λ_t) and the debt ratio (denoted by d_t). This system is described as follows:

$$\frac{\partial \omega_t}{\partial t} = \omega_t(\phi(\lambda_t) - \alpha) \tag{8}$$

$$\frac{\partial \lambda_t}{\partial t} = \lambda_t\left(\frac{\kappa(1-\omega_t - rd_t)}{\upsilon} - \alpha - \beta - \delta\right) \tag{9}$$

$$\frac{\partial d_t}{\partial t} = d_t\left(r(1-\Delta) - \frac{\kappa(1-\omega_t - rd_t)}{\upsilon} + \delta\right) + \kappa(1-\omega_t - rd_t) - (1-\omega_t)(1-\Delta) \tag{10}$$

where α is constant growth rates of the labor productivity, β is the constant growth rates of the labor force, δ is the depreciation rate, υ is the capital-to output ratio, r is the real interest rate and Δ is the newly defined share of profits distributed to

shareholders. Function $\phi(\lambda_t) = \gamma + \rho(1-\lambda_t)^{-2}$ is the real short-run Phillips curve and $\kappa(1-\omega_t - rd_t) = k_1 e^{k_2(1-\omega_t - rd_t)}$ is an investment function.

3.3 Cross Correlation Function

The cross-covariance function measures how two time series are related by assessing how changes in one series relate to past values of the other over different time lags. For two series X_t and Y_t, the cross-covariance function at lag h is defined as:

$$Cov(X_t, Y_{t-h}) = E\left((X_t - \mu_x)(Y_{t-h} - \mu_y)\right) \tag{11}$$

where μ_x and μ_y are the means of X_t and Y_t, respectively.

The closely related cross-correlation function (CCF) is a statistical tool utilized to measure the linear dynamic dependence of two series. It is essentially the cross-covariance normalized by the product of the standard deviations of the two series. Mathematically, the CCF at lag h is given by:

$$Corr(X_t, Y_{t-h}) = \rho_{xy}(h) = \frac{E\left((X_t - \mu_x)(Y_{t-h} - \mu_y)\right)}{\sigma_x \sigma_y} \tag{12}$$

where σ_x and σ_y are the standard deviations of X_t and Y_t, respectively.

In practice, these quantities are estimated using sample means and variances. The sample CCF at lag h is computed as:

$$r_{xy}(h) = r(x_t, y_{t-h}) = \frac{\sum_{t=1}^{n-h}(x_t - \bar{x})(y_{t-h} - \bar{y})}{\sqrt{\sum_{t=1}^{n}(x_t - \bar{x})^2 \sum_{t=1}^{n}(y_t - \bar{y})^2}} \tag{13}$$

where $\bar{x}$ and $\bar{y}$ denote the sample means of x_t and y_t. The sample CCF values lie between -1 and 1, indicating the strength and direction of the linear relationship at each lag.

3.4 Deep Neural Networks and Automatic Differentiation

Deep learning, a subsection of machine learning, is the cutting-edge technology used to enhance data analysis and predictive capabilities. It employs multi-layer artificial neural networks to construct algorithms that can extract meaningful patterns and features from extensive data sets effectively. This approach has been effectively utilized in diverse areas, including image process and speech recognition, natural language process (NLP), gaming, and self-driving vehicles. Deep learning excels at managing complex, high-dimensional data. The main benefit of deep learning models is their capability to automatically identify features from data, minimizing the need for manual feature engineering and greatly enhancing the overall performance of the model.

3.4.1 Perceptron and Deep Neural Networks

The perceptron is a building block of artificial neural network laying the groundwork for more complex networks. It was devised in the 1960s by scientist Frank Rosenblatt [38]. The perceptron comprises a single neuron and an activation function. The input of the perceptron is denoted as x, the corresponding weights are denoted as w, the associated bias is denoted as b, and f represents the activation function. As depicted in Figure 3.1, before applying f, the perceptron takes the input x, computes the weighted sum, and adds the bias.

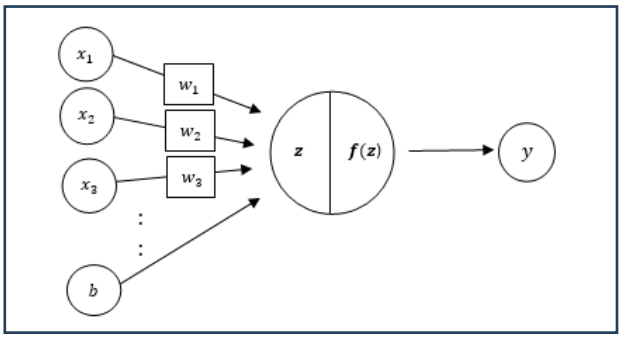

Figure 3.1 Structure of Perceptron

Hence, for input x, our perceptron makes the following predictions:

$$y = f(z) = f(wx^T + b) = f(\sum_{i=1}^{n} x_i w_j + b) \tag{14}$$

The process for building more complex neural networks is not significantly different. A feed-forward neural network (FFNN) or multi-layer perceptron neural network is an extension of the basic perceptron, consisting of multiple layers of neurons, with each layer fully connected to the next. FFNNs are called "feed-forward" because inputs are processed forward through the network without any loops or cycles. They can easily solve more complex problems than a single-layer perceptron due to their depth and complexity. The architecture of a FFNN is depicted in Figure 3.2. In a neural network, a layer comprises individual neurons that process inputs separately and generate a sole output. When a neural network is referred to as “deep,” it means that there are multiple layers between the input and output layer. Since each neuron has a weight vector used to compute its output, a “weight matrix” is created to mathematically describe the collection of weight vectors in each layer.

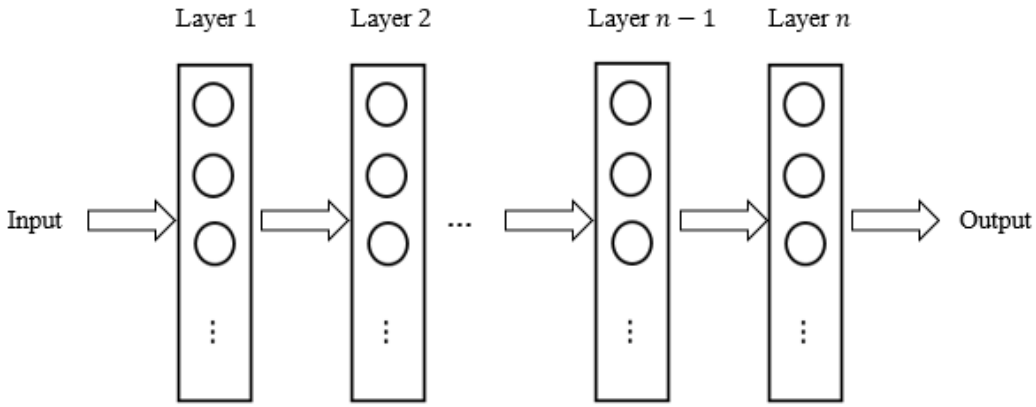

Figure 3.2 Structure of the FFNN

3.4.2 Training and Automatic Differentiation

Training a neural network is quite intricate, involving initialization, forward and backward passes, loss calculation, and parameter updates. The main objective is to adjust the network's weights and biases to minimize prediction errors. This enables the network to learn from the training data and generalize its findings. This training is crucial as it allows the model to improve its accuracy and generalize new situations by capturing complex patterns, making accurate predictions, and adapting to new information. Properly trained neural networks can achieve state-of-the-art performance.

The gradient descent algorithm is a method used to iteratively optimize and find the local minimum of a function. This algorithm is crucial for learning from training data as it assists in determining the set of weights that minimally correspond to prediction errors by minimizing the loss function of the network. The loss function assesses the difference between the predicted outputs and the actual outputs, and the goal of training is to reduce this difference. The steps of the Gradient Descent Algorithm are as follows:

1. Parameter Initialization: Begin with initial values for the parameters.

2. Gradient Computation: Determine the gradient of the loss function concerning each parameter.

3. Parameter Adjustment: Adjust the parameters in the direction opposite to the gradient to decrease the loss.

4. Iteration: Repeat the process until the loss converges or a maximum number of iterations is reached.

Backpropagation is an essential technique for training artificial neural networks. It efficiently calculates the gradient of the loss function for each weight by propagating the error backward through the network. Automatic Differentiation (AD) efficiently computes gradients by numerically evaluating the derivative of a function specified

by a computer program. AD exploits the differentiability of every computer operation, from simple arithmetic to complex functions, and automates the calculation of derivatives using the chain rule. Figure 3.3 offers an accurate representation of how AD computes $\partial E/\partial \omega_1$ and $\partial E/\partial \omega_2$.

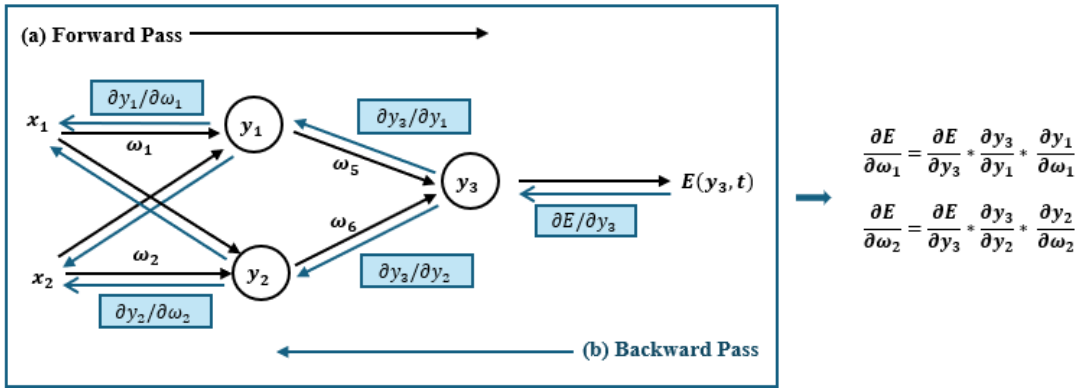

Figure 3.3 Computational Graph of AD

The backpropagation is utilized to enhance prediction accuracy in artificial neural networks by modifying the model's weights and biases. This adjustment is accomplished by transmitting the error from the output layer back to the input layer. The process begins with a forward pass where each neuron computes the weighted sum of its inputs, adds a bias term, and then applies an activation function to the result. This process continues through the network, layer by layer, until the final output values are produced. Afterward, the loss function is calculated. During the backward pass, the gradient which is the derivative of the loss function with respect to parameters is computed using the chain rule of calculus. Once these gradients are computed, the weights and biases are updated using them to minimize the loss. These updates are typically done using the gradient descent algorithm or its variations, which adjust the weights in the opposite direction of the gradient.

3.5 Long Short-Term Memory Networks

The concept of Long Short-Term Memory (LSTM) networks was presented by Hochreiter and Schmidhuber in 1997 [14]. These networks have revolutionized sequence modeling by overcoming the constraints of traditional RNNs and facilitating the successful acquisition of long-term dependencies. Traditional RNNs maintain a hidden state that is updated at each time step to handle sequences. However, they struggle with long-term dependencies due to the vanishing gradient problem, where gradients decrease exponentially as they are backpropagated through time. Therefore, RNNs have difficulty learning long-term dependencies and retaining information over long sequences. LSTMs solve this problem by introducing a more complex unit structure that includes gates to control the flow of information. Their capacity to retain data across long sequences and address the issue of the vanishing gradient gives them significant utility for various areas, including NLP, speech recognition, image process, time series prediction, and video analysis.

An LSTM network consists of LSTM units, each with a cell state and three gates: the input gate, the forget gate, and the output gate. These gates of the LSTM units regulate the flow of information into, out of, and within the units.

1. Cell State: The cell state acts as a memory, preserving information across various time steps. It is modified by the gates to preserve or discard information.
2. Input Gate: It controls the flow of new information that enters the cell state. It determines which values from the input should be utilized to update the cell state.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{15}$$

   where i_t represents the input gate, h_{t-1} represents the output of the previous LSTM block, x_t represents the input at time step t, b_i is the bias for the input gate, W_i is the weight for the input x_t , U_i is the weight for the output of the previous LSTM block h_{t-1} and σ represents the sigmoid function.
3. Forget Gate: It decides which information from the cell state to ignore, allowing the LSTM to discard irrelevant information.

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \quad (16)$$

where f_t represents the forget gate, b_f is the bias for the forget gate, W_f is the weight for the input x_t , U_f is the weight for the output of the previous LSTM block h_{t-1} and σ represents the sigmoid function.

4. Output Gate: It governs the output of the LSTM unit. It decides which parts of the cell state should be transmitted to the next time step as the hidden state.

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (17)$$

where o_t represents the output gate, b_o is the bias for the output gate, W_o is the weight for the input x_t , U_o is the weight for the output of the previous LSTM block h_{t-1}.

5. Cell Update: It is refreshed using the information from the input gate and the forget gate.

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (18)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \quad (19)$$

where $\tilde{c}_t$ represents the candidate for the cell state at time step t , c_t is the cell state (memory) at time step t, W_c is the weight for input x_t , U_c is the weight for the output of the previous LSTM block h_{t-1}, b_c is the bias for the cell state, tanh is the hyperbolic tangent function and "$\circ$" represents elementwise multiplication of the vectors.

6. Hidden State Update: The hidden state is updated based on the cell state and the output gate.

$$h_t = o_t \circ \tanh(c_t) \quad (20)$$

At each time step, an input x_t and the previous hidden state h_{t-1} are received by the LSTM cell. The current hidden state h_t and the current output y_t are then produced

by the LSTM cell. Sigmoid activation functions are utilized by the input, forget, and output gates. These functions yield values within the range of 0 to 1. These values are subsequently multiplied by the input and the previous hidden state to determine the amount of information that enters the cell state, the amount that is removed from the cell state, and the amount that is output from the cell state. The cell state is then passed through a tanh function to generate the hidden state. Following this, the hidden state undergoes processing through a fully connected layer featuring a sigmoid or softmax activation function to generate the output. This functionality allows LSTM networks to selectively determine the information to store, discard, and output. The visual representation of the memory block in an LSTM is illustrated in Figure 3.4.

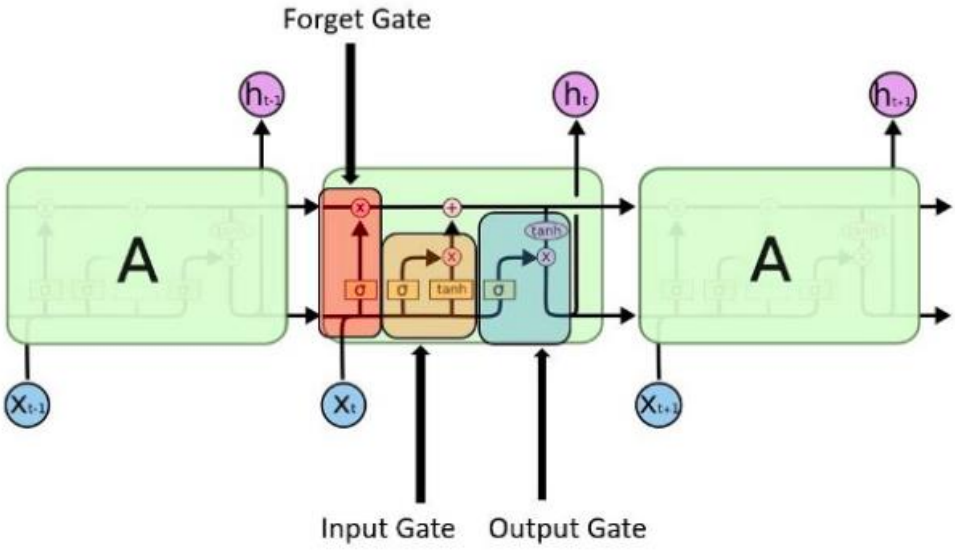

Figure 3.4 Memory Block in LSTM [37]

3.6 Physics Informed Neural Networks

PINNs directly incorporate physical laws or differential equations into the training process of neural networks. This integration ensures that the predictions align with established scientific principles while learning from data. By including known deterministic relationships between variables in neural networks, PINNs use this extra information to improve the prediction and forecast performance of the models.

3.6.1 Architecture

Training a deep learning algorithm may seem difficult or even impractical to accurately identify a nonlinear map from a limited number of input and output data pairs. Achieving model accuracy with complex and high-dimensional data may pose challenges. Deep learning algorithms have demonstrated their capability in approximating complex nonlinear mappings. However, these training processes necessitate a substantial amount of training data to generalize well for unseen data. If the available data is limited or issues related to high dimensionality arise, training the model becomes exceedingly challenging.

In the field of biology or physics, it is essential to integrate existing knowledge to enhance the precision of the model. This existing knowledge may take the form of fundamental physical laws, empirically verified principles, or other expertise within the specific domain. By embedding this organized information into the learning algorithm, the information content of the available data can be enhanced, allowing the model to rapidly converge towards the correct solution and exhibit effective generalization, even in cases where only a small number of training examples are accessible. Within this framework, Raissi et al. [35] has introduced a machine learning approach that merges the capabilities of deep neural networks with the fundamentals of physics, known as PINNs. Their explanation of the method and its applications has laid the groundwork for recent advancements in using neural networks and differential equations together. The key feature of this technique is to perform regression and model estimation by leveraging the properties of both differential equations and neural networks. In the field of physical modeling, it is typical to describe the dynamics of a system using a series of differential equations. PINNs attempt to leverage the fundamental features of a solution to a differential equation to efficiently regulate a neural network model. This is accomplished by embedding the mathematical model into the network and enhancing the loss function by adding a residual term that is derived from the governing equation, effectively serving as a penalizing factor to confine the acceptable solution space. The motivation

for this varies, but it does introduce some bias to models that would typically have been unbiased. In the physics-informed approach, the differential equation serves as the regularization cost for the model's parameters. A typical architecture of PINNs is depicted in Figure 3.5. In Figure 3.5, a fully connected neural network uses time and space coordinates (t, x) as inputs to approximate the multi-physics solutions. AD is used to compute the derivatives with respect to the inputs, which are then employed to define the residuals of the governing equations within the loss function. This function usually consists of several terms, each weighted by different coefficients. Both the neural network parameters (θ) and the unknown parameters of the PDEs (λ) can be learned simultaneously by minimizing the loss function.

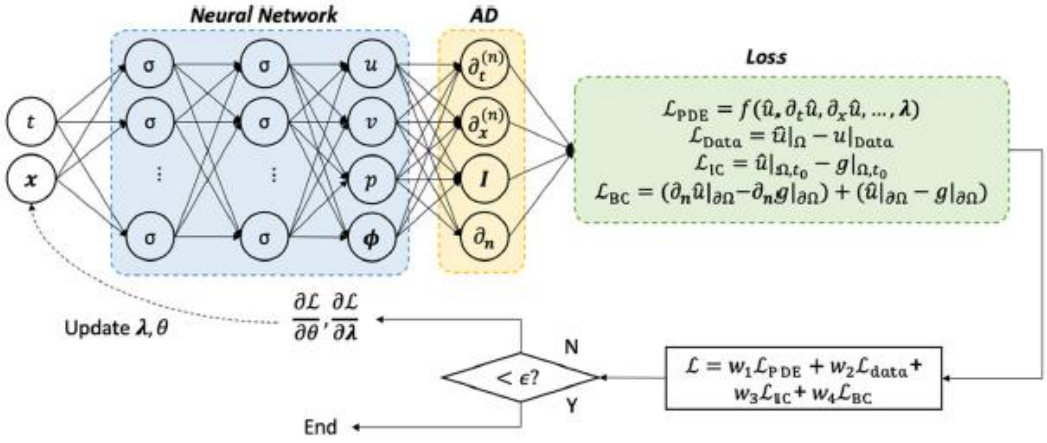

Figure 3.5 Architecture of PINNs [7]

In PINNs approach, the loss function is designed in a way that the neural network not only fits the data but also satisfies the physical laws and constraints represented by the differential equation and the boundary/initial conditions. This means that PINNs incorporate physics into the learning process. The parameterized and nonlinear partial differential equations (PDEs) of the general form:

$$u_t + \mathcal{N}[u;\lambda] = 0 \quad , \quad x\epsilon\Omega \quad , \quad t\epsilon[0,T] \tag{21}$$

In here, u_t denotes the *t* derivative of $u(t,x)$ which denotes the hidden/latent solution to the equation, which refers to the unknown function that we are trying to find or approximate. It is a function of two variables, time (t) and position (x). $\mathcal{N}[.]$ represents a nonlinear differential operator parameterized by λ. An operator is a mathematical symbol or function that operates on a function to produce another function. In this case, $\mathcal{N}[.]$ is a nonlinear operator that acts on the function $u(t,x)$ to yield a new function. Ω denotes a subset of R^D. This setup covers a wide range of problems in mathematical physics including kinetic equations, conservation laws, advection–diffusion–reaction systems and diffusion processes. Raissi et al. [35] focus on solving two different problems. The first one is data driven solution of PDEs, which refers to *"Given fixed model parameters λ what can be said about the unknown hidden state u(t, x) of the system?".* The second one is data driven discovery of PDEs, which refers to "*What are the parameters λ that best describe the observed data*?".

3.6.2 Data-Driven Solutions of Partial Differential Equations

The parametrized and nonlinear PDEs of the general form is considered:

$$u_t + \mathcal{N}[u] = 0 \quad , \quad x\epsilon\Omega \quad , \quad t\epsilon[0,T] \tag{22}$$

where u_t denotes the *t* derivative of $u(t,x)$ which denotes the hidden/latent solution, $\mathcal{N}[.]$ is a nonlinear operator and Ω is a subset of R^D. We define $f(t,x)$ to be given by the left-hand side of equation given above:

$$f := u_t + \mathcal{N}[u] \tag{23}$$

and proceed by approximating $u(t,x)$ by a deep neural network. This assumption along with the above equation results in a *physics informed neural network* $f(t,x)$. This network can be derived by applying the chain rule for differentiating compositions of functions using AD and has the same parameters as the network

representing $u(t,x)$, albeit with different activation functions due to the action of the differential operator $\mathcal{N}$. The shared parameters between the neural networks $u(t,x)$ and $f(t,x)$ can be learned by minimizing the mean squared error (MSE) loss. The loss function over data for this network can be realized as the combined MSE loss:

$$MSE_{pinn} = MSE_u + MSE_f \tag{24}$$

where

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2 \tag{25}$$

and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} \left|f(t_f^i, x_f^i)\right|^2 \tag{26}$$

where the sets $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ denote the initial and boundary training data on $u(t,x)$. $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ specifies the collocations points for $f(t,x)$. The loss MSE_u corresponds to the initial and boundary training data while MSE_f enforces the structure imposed by equation f at a finite set of collocation points.

3.6.3 Data-Driven Discovery of Partial Differential Equations

We consider parametrized and nonlinear PDEs of the general form:

$$u_t + \mathcal{N}[u;\lambda] = 0 \quad , \quad x\epsilon\Omega \quad , \quad t\epsilon[0,T] \tag{27}$$

where u_t denotes the *t* derivative of $u(t,x)$ which denotes the hidden/latent solution, $\mathcal{N}[.]$ is a nonlinear operator parameterized by λ and **Ω** is a subset of R^D. We define $f(t,x)$ to be given by the left-hand side of equation given above:

$$f := u_t + \mathcal{N}[u;\lambda] \tag{28}$$

We proceed by approximating $u(t,x)$ by a deep neural network. This assumption along with the above equation results in a *physics informed neural network* $f(t,x)$. This network can be derived by applying the chain rule for differentiating compositions of functions using AD. The parameters λ of the nonlinear differential operator as well as the parameters of the neural networks can be learned by minimizing the MSE loss. The loss function used for this network corresponds to that defined in Equations (24 - 26).

3.7 Prior-Informed Multivariate LSTM (PIM-LSTM)

In this study, we propose the Prior-Informed Multivariate LSTM (PIM-LSTM), a novel deep learning architecture designed to incorporate prior knowledge constraints into multivariate time-series forecasting. The proposed PIM-LSTM model, inspired by Physics-Informed Neural Networks (PINNs), leverages prior knowledge constraints and incorporates cross-correlation function (CCF) constraints to improve multivariate time-series forecasting.

A sequence of inputs is fed into a LSTM. The LSTM processes these inputs over time, capturing both short-term and long-term dependencies via its hidden and cell states. At the final step, the network produces a prediction, which is then compared to the true target. The first loss component is the standard Mean Squared Error on the training data. It measures how closely LSTM's predictions align with the ground truth values. For three output model, when output is $\vec{X} = \{x, y, z\}$, MSE is calculated as follows:

$$MSE_{train} = \frac{1}{3n}\left(\sum_{t=1}^{n}(\hat{x}_t - x_t)^2 + \sum_{t=1}^{n}(\hat{y}_t - y_t)^2 + \sum_{t=1}^{n}(\hat{z}_t - z_t)^2\right) \qquad (29)$$

The second loss component is obtained from prior knowledge or constraints. For three output model, when output is $\vec{X} = \{x, y, z\}$, MSE is calculated as follows:

$$MSE_{prior} = \frac{1}{3n}(\sum_{t=1}^{n}\left(\hat{x}_t - f(x_t)\right)^2 + \sum_{t=1}^{n}\left(\hat{y}_t - f(y_t)\right)^2 + \sum_{t=1}^{n}\left(\hat{z}_t - f(z_t)\right)^2 \quad (30)$$

where $\hat{x}_t$, $\hat{y}_t$ and $\hat{z}_t$ are the predictions obtained from the model and f is the prior function, reflecting the expected or theoretically derived behavior. This could be a known relationship from physics, an empirical formula from domain experts, or any other expression encoding constraints or knowledge relevant to the problem.

The final loss is CCF loss, which typically aims to preserve certain correlation structures in time-series data. The CCF loss measures how closely the predicted series' cross-correlation function matches either the observed data's cross-correlation or a desired correlation pattern. Symbolically, it can be written as:

$$\mathcal{L}_{CCF} = \frac{1}{2 \times \ell \times k}\sum_{h=1}^{\ell}\sum_{(x,y)\in S}\left(r(\hat{x}_t, \hat{y}_{t-h}) - r(x_t, y_{t-h})\right)^2 \quad (31)$$

where $r(x_t, y_{t-h}) = \frac{\sum_{t=1}^{n-h}(x_t - \bar{x})(y_{t-h} - \bar{y})}{\sqrt{\sum_{t=1}^{n}(x_t - \bar{x})^2 \sum_{t=1}^{n}(y_t - \bar{y})^2}}$.

ℓ denotes the number of lags, while k refers to the number of output variables. $r(\hat{x}_t, \hat{y}_{t-h})$ denotes the cross-correlation function at lag h of predicted $\hat{x}_t$ and predicted $\hat{y}_{t-h}$. $r(x_t, y_{t-h})$ denotes the cross-correlation function at lag h of actual x_t and actual y_{t-h}. S is the set of output variable pairs. For example, for the New Keynesian model, the output pairs are $S = \{(\pi, i), (\pi, y), (i, \pi), (i, y), (y, \pi), (y, i)\}$. For the DAGKM model, the output pairs are $S = \{(\omega, \lambda), (\omega, d), (\lambda, \pi), (\lambda, d), (d, \omega), (d, \lambda)\}$. By minimizing $\mathcal{L}_{CCF}$, the model learns to generate forecasts whose interdependence or temporal correlations more closely match those in the true data (or match specified domain expectations).

All these terms MSE_{train}, MSE_{prior} and $\mathcal{L}_{CCF}$ are combined into a single total loss function. A typical form is:

$$\mathcal{L}_{total} = MSE_{train} + \alpha_1 MSE_{prior} + \alpha_2 \mathcal{L}_{CCF} \quad (32)$$

where α_1 and α_2 are weighting coefficients that balance the importance of fitting the training data, adhering to priors, and matching the desired correlation structure. The gradient of this total loss concerning the LSTM parameters is computed (via backpropagation through time) and used to update the model weights. By jointly optimizing these three objectives, the model learns to produce forecasts that are accurate. The architecture of the PIM-LSTM is depicted in Figure 3.6 and 3.7 for the New Keynesian model and the DAGKM model.

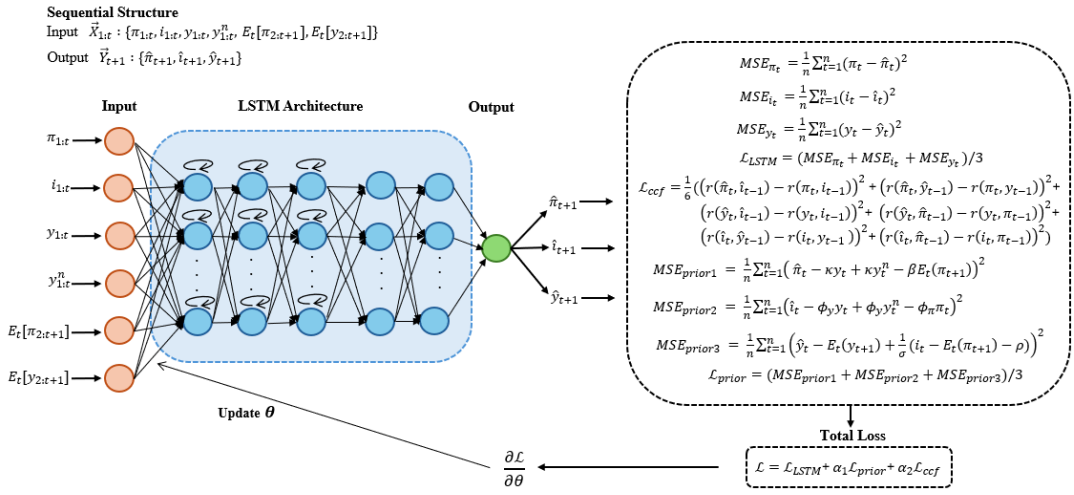

Figure 3.6 Architecture of PIM-LSTM Model for the New Keynesian Model

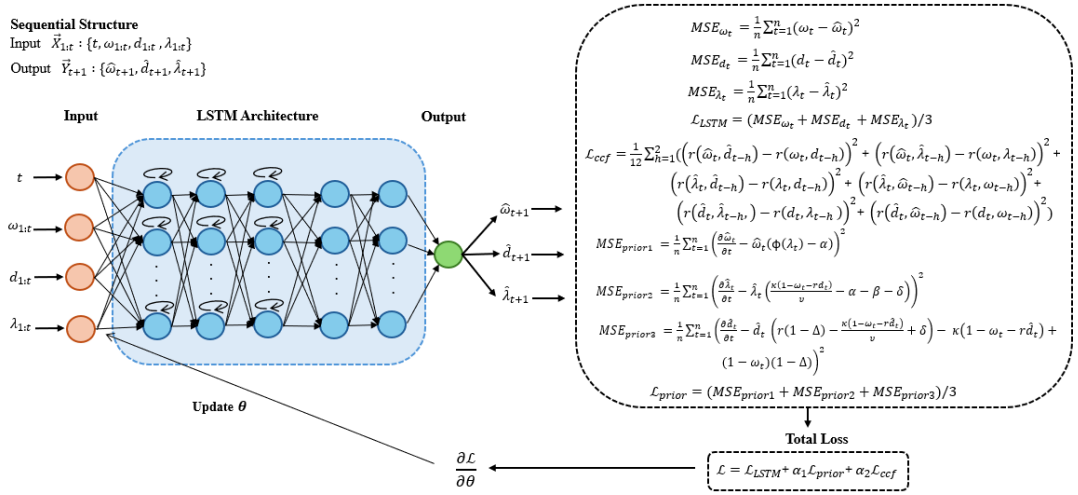

Figure 3.7 Architecture of PIM-LSTM Model for the DAGKM

3.8 Performance Metrics

When forecasting multivariate time series data, it is important to check how accurate the forecasts are. There are different ways to measure how well a forecasting model works and each method has its own advantages. Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Scaled Error (MASE) are commonly used metrics. RMSE quantifies the average magnitude of errors in a set of predictions. Its sensitivity to larger errors makes RMSE particularly advantageous in situations where significant discrepancies require penalization. RMSE can be determined using the following formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2} \tag{33}$$

where y_t is the actual value at time t, $\hat{y}_t$ is the forecasted value at time t and n is the number of observations.

MAE provides a straightforward way to assess the average size of errors in a group of predictions, irrespective of whether they are positive or negative. Compared to RMSE, MAE is less affected by outliers. The calculation for MAE can be expressed with the following formula:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t| \tag{34}$$

where y_t is the actual value at time t, $\hat{y}_t$ is the forecasted value at time t and n is the number of observations.

MASE is a metric that evaluates the accuracy of forecasts across various datasets by normalizing errors relative to a naive estimate, typically derived from the mean of previous observations.

$$MASE = \frac{MAE}{MAE_{naive}} = \frac{\frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|}{\frac{1}{n-1}\sum_{t=2}^{n}|y_t - y_{t-1}|} \tag{35}$$

CHAPTER 4

DATA ANALYSİS

In this study, PIM-LSTM model is applied for multivariate time series forecasting across three different datasets from economics. Its performance is thoroughly assessed by comparing it with various well-established benchmark models.

4.1 Dataset for the New Keynesian Model for Turkiye

The New Keynesian model for Turkiye is employed to apply the PIM-LSTM method. The model is composed of several equations for the inflation rate, output gap and nominal interest rate. The consumer price index (CPI), real gross domestic product (GDP), the short-term interest rate, the inflation forecast, and the real GDP forecast are used. We utilize a quarterly dataset of these variables from 2002Q4 to 2021Q3 for Turkiye. The real GDP and the CPI are sourced from IMF International Financial Statistics. The inflation forecast, the real GDP forecast and the short-term interest rate are retrieved from OECD Data Explorer.

Before conducting exploratory data analysis, some transformations on the dataset are implemented. Initially, following the recommendations of Pfeifer [33], these calculations are performed:

$$\begin{aligned} &\text{i.} \quad y_t^{obs} = \log\left(y_t^{data}\right) \\ &\text{ii.} \quad \pi_t^{obs} = \log\left(\frac{cpi_t^{data}}{cpi_{t-1}^{data}}\right) - mean\left(\log\left(\frac{cpi_t^{data}}{cpi_{t-1}^{data}}\right)\right) \\ &\text{iii.} \quad i_t^{obs} = \log\left(1 + \frac{i_t^{data}}{4\times100}\right) - mean\left(\log\left(1 + \frac{i_t^{data}}{4\times100}\right)\right) \end{aligned} \tag{36}$$

Secondly, the real GDP is seasonally adjusted by using Census X-13 procedure since the real GDP shows seasonal pattern. Then, two-sided Hodrick-Prescott (HP) filter

is utilized to extract the trend component of seasonally adjusted real GDP. The Hodrick-Prescott (HP) filter is a technique utilized to separate a time series into its trend and cyclical components. It is commonly employed in macroeconomics to decompose a real GDP y_t into potential output (trend component) and output gap (cyclical component) by minimizing this function:

$$\sum_{t=1}^{T}(y_t - \tau_t)^2 + \lambda \sum_{t=2}^{T-1}\left((\tau_{t+1} - \tau_t) - (\tau_t - \tau_{t-1})\right)^2 \quad (37)$$

where $y_t = \tau_t + c_t$, τ_t is trend component and c_t is cyclical component. Potential output data obtained from HP filter is added to the dataset. Additionally, potential output, inflation forecasts, and GDP forecasts are considered exogenous variables.

4.1.1 Exploratory Data Analysis

Table 4.1 provides descriptive statistics of key macroeconomic indicators for Turkiye, specifically the CPI inflation rate, the short-term interest rate, the real GDP, the potential output, the inflation forecast, and the GDP forecast. The CPI inflation rate has a mean close to zero (-0.0004) with a range from -0.0185 to 0.0913. Half of the CPI inflation rate values fall between -0.0103 and 0.0005. The short-term interest rate has a mean of -0.0008. Half of the short-term interest rate values are between -0.0078 and 0.0019. Similarly, the real GDP and the potential output values are closely aligned, with means of 5.5585 and 5.5580, respectively. The CPI inflation rate, the short-term interest rate, the real GDP, the potential output are probably symmetrically distributed since their means and median values are close. The inflation forecast and the GDP forecast exhibit the highest dispersion, as indicated by their wide ranges (from 4.2894 to 81.0785 and -36.1690 to 82.9082, respectively) and large standard deviations (18.4976 and 11.9554). In contrast, the interest rate, inflation rate, real GDP, and potential output exhibit lower variability, as indicated by relatively small standard deviations that denote greater stability.

Table 4.1 Descriptive Statistics of Turkiye's Macroeconomic Series

	Inflation Rate	Interest Rate	Real GDP	Potential Output	Inflation Forecast	GDP Forecast
Minimum	-0.0185	-0.0124	5.3037	5.3152	4.2894	-36.1690
Q_1	-0.0103	-0.0078	5.4413	5.4410	8.0459	1.3710
Median	-0.0053	-0.0036	5.5704	5.5627	9.4801	5.9014
Mean	-0.0004	-0.0008	5.5585	5.5580	17.3075	5.8449
Standard Dev.	0.0181	0.0108	0.1385	0.1379	18.4976	11.9554
Q_3	0.0005	0.0019	5.6734	5.6744	14.5977	9.8588
Maximum	0.0913	0.0343	5.7887	5.7921	81.0785	82.9082

There is no missing data in any of the series. The time series graphs of Turkiye's macroeconomic series are shown above. Figure 4.1 illustrates the changes in the CPI inflation rate from 2002Q4 to 2021Q3. The time series plot of the CPI inflation rate does not exhibit any seasonal behavior and displays only slight fluctuations over time. Initially, there is a sharp decline, followed by frequent ups and downs around the zero line. After 2012, there appears to be a slight upward trend. Figure 4.2 shows how the interest rate changed from 2002Q4 to 2021Q3. The interest rate shows no seasonal pattern and follows a decreasing trend from 2002 to 2013. Although there is a sudden decline between 2019 and 2020, there has been a gradual increase in the short-term interest rate since 2013. Figure 4.3 illustrates the changes in real GDP from 2002Q4 to 2021Q3. The real GDP shows no seasonal behavior and demonstrates an overall upward trend during this period, despite occasional short-term declines.

By examining sample ACF and sample PACF in Figures 4.4, 4.5, and 4.6, inflation rate and interest rate could be stationary since their ACFs show exponential decay, their PACFs cut off after lags 1 and 2. However, the ACF of the real GDP shows slow decay, indicating that it is non-stationary.

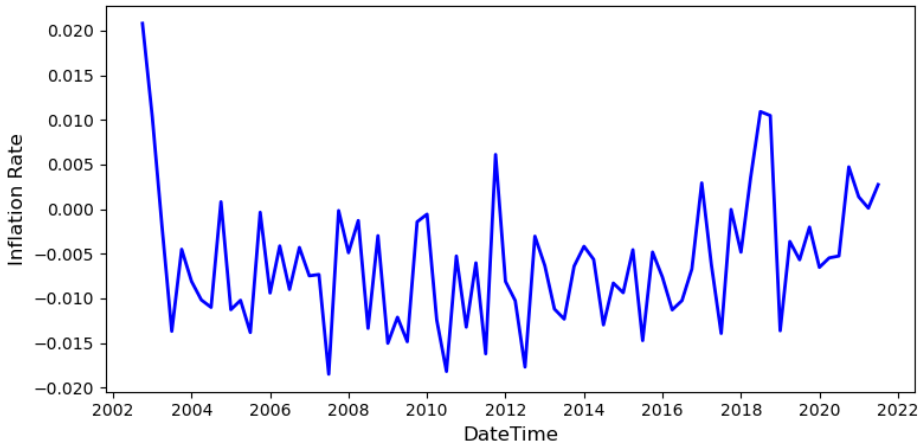

Figure 4.1 The Time Series Plot of Turkiye's Inflation Rate

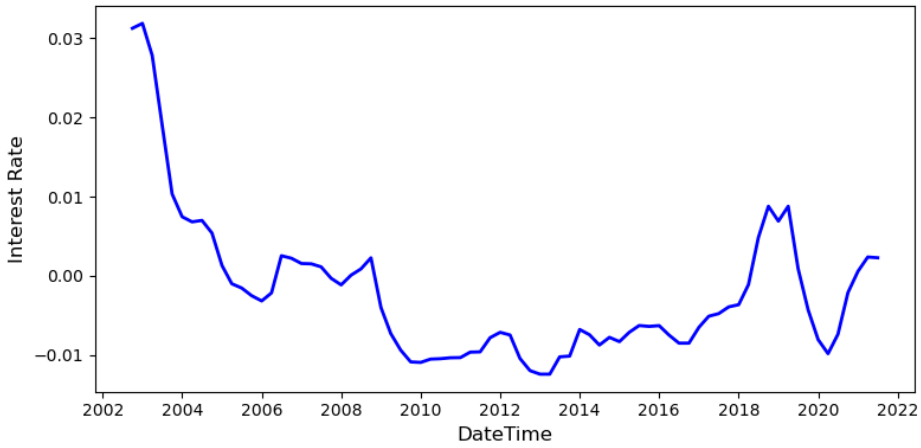

Figure 4.2 The Time Series Plot of Turkiye's Interest Rate

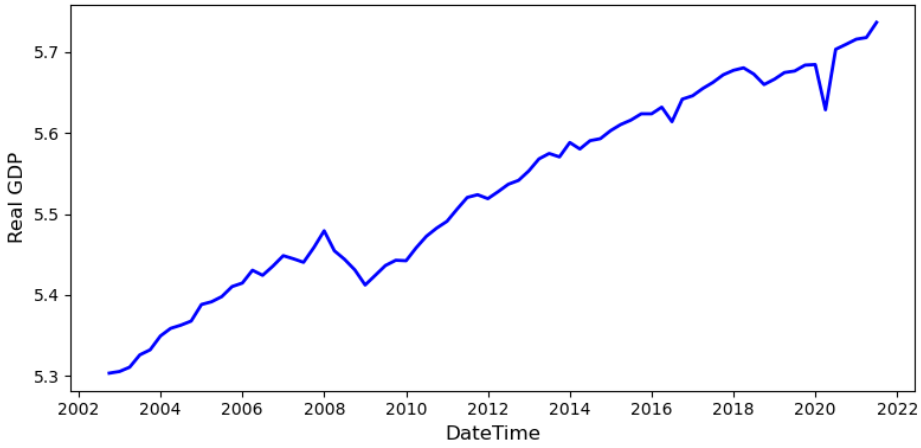

Figure 4.3 The Time Series Plot of Turkiye's Real GDP

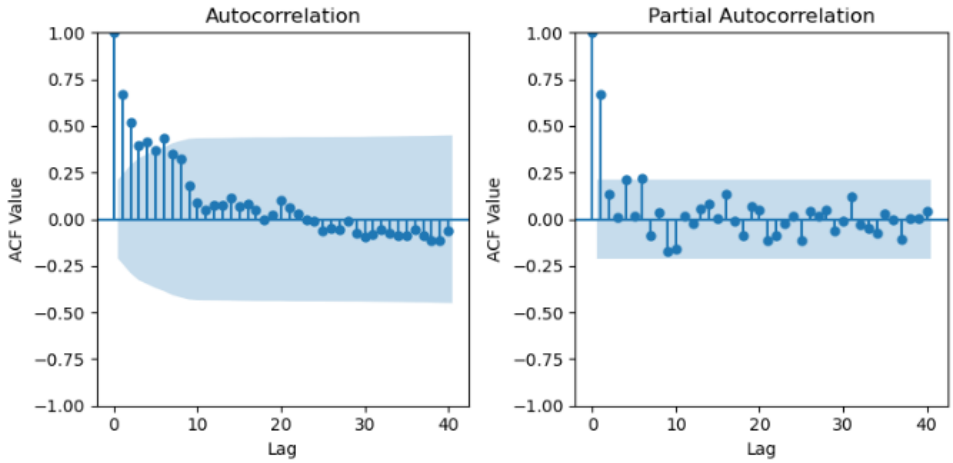

Figure 4.4 The sACF and sPACF of Turkiye's Inflation Rate

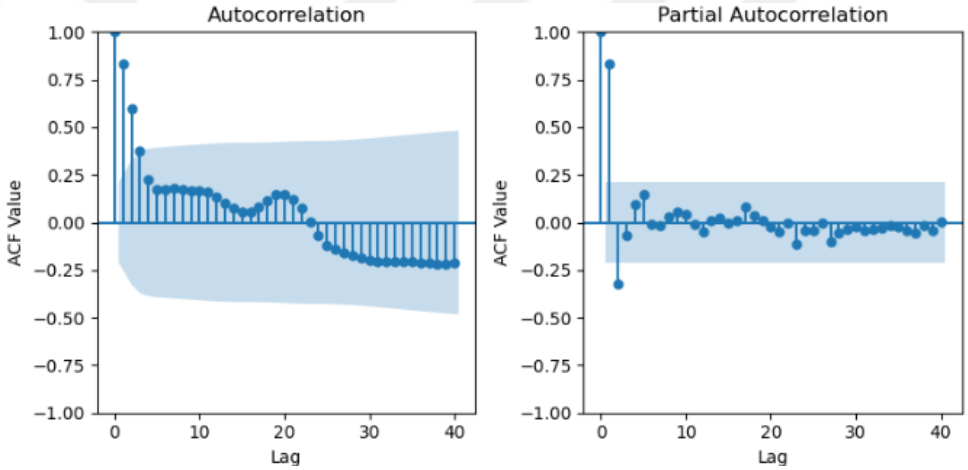

Figure 4.5 The sACF and sPACF of Turkiye's Interest Rate

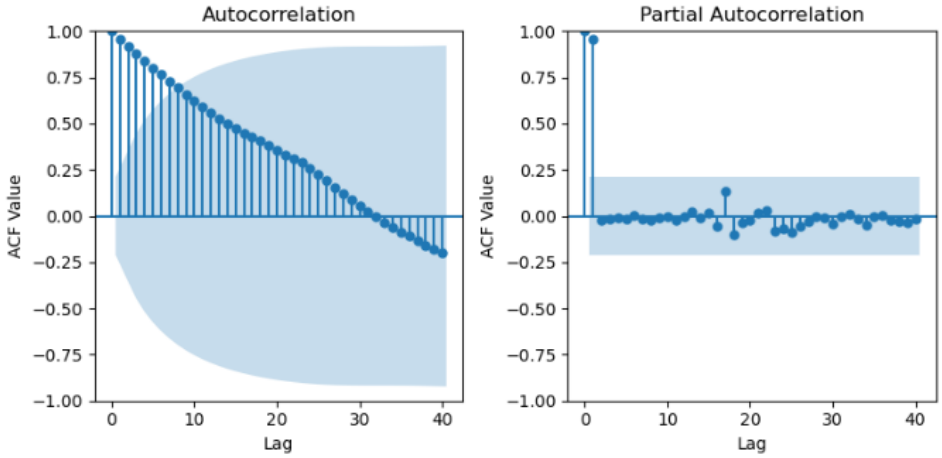

Figure 4.6 The sACF and sPACF of Turkiye's Real GDP

Augmented Dickey-Fuller (ADF) Test

The ADF test is a statistical method used to determine if a time series is stationary or not. The hypothesis of the test is:

- H_0 : The time series has a unit root (is non-stationary)
- H_1 : The time series is stationary

ADF test results indicate that inflation rate, interest rate and real GDP have unit roots, indicating nonstationarity.

Table 4.2 ADF test for Inflation Rate

Test Statistic	0.2193
Number of Lags Used	5
P-value	0.9733

Table 4.3 ADF test for Interest Rate

Test Statistic	−1.4759
Number of Lags Used	4
P-value	0.5454

Table 4.4 ADF test for Real GDP

Test Statistic	−0.9467
Number of Lags Used	1
P-value	0.7722

Toda-Yamamoto Causality Test

In multivariate time series analysis, multiple interdependent variables are dealt with. Before conducting models, it is necessary to examine causal relationships. For this purpose, the Toda-Yamamoto Causality test is the most appropriate method to employ since the series exhibits a unit root. This test is a recognized statistical

technique that facilitates the analysis and identification of causal relationships between variables. The hypothesis of the test:

- H_0 : The series x does not granger cause the series y.
- H_1 : The series x granger causes the series y.

Toda-Yamamoto Causality test results given in Table 4.5 indicate several significant relationships between the variables. The short-term interest rates and the real GDP granger cause the CPI inflation rate. The CPI inflation rate and the real GDP granger cause the short-term interest rate. The short-term interest rate granger causes the real GDP whereas the CPI inflation rate does not granger cause the real GDP. These results suggest the presence of causal relationships among certain variables, indicating that multivariate time series analysis is applicable to this dataset.

Table 4.5 Toda-Yamamoto Causality Test Result

X => Y			**P value**
Interest Rate	=>	Inflation Rate	0.018
Interest Rate	=>	Real GDP	0.000
Inflation Rate	=>	Interest Rate	0.000
Inflation Rate	=>	Real GDP	0.935
Real GDP	=>	Interest Rate	0.032
Real GDP	=>	Inflation Rate	0.000

4.1.2 Data Preprocessing

Data preprocessing is the set of operations performed on raw data before applying a machine learning or deep learning model. It typically includes steps such as cleaning, normalization or scaling and structuring data into a suitable format. By ensuring data quality and consistency, preprocessing helps models learn more effectively and converge faster, ultimately leading to more reliable results.

Min-Max normalization is a useful method because it scales data to a fixed range, usually between 0 and 1, while keeping the original data distribution intact. This approach makes sure that all features influence the training process equally, avoiding

issues where features with larger scales might have too much impact on the model. By reducing the range of the data, Min-Max normalization also helps improve the stability and speed of many optimization algorithms, which can lead to better performance and reliability in deep learning models.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{38}$$

This study employs supervised deep learning models to analyze time series data. The input data is converted into sequences for the models to learn from previous input values and predict future output values. The generation of sequences is based on three key principles: input sequence length, sliding window, and output sequence length. Input sequence length refers to the number of time steps in the input data that the model utilizes to forecast the associated output. The sliding window technique creates multiple sequences of the same length by sliding the sequence to '*n*' steps. Output sequence length, also known as prediction horizon, determines the number of future values that the model forecasts based on the input sequence provided. In this study, input sequences are generated using a sliding window approach with a stride of one quarter, which refers to the step size used when shifting the sliding window to create new input-output sequences. Each output sequence consists of three values, each representing a one-step-ahead forecast based on past values of the series. For example, to forecast macroeconomic data for the upcoming quarter using data from the previous 4 quarters, we first organize the data from those 4 quarters into a sequence. The goal is to predict the values for the 5th quarter. After that, we shift the time window by one quarter to create a new sequence, now consisting of data from the 2nd to the 5th quarter and aim to predict the values for the 6th quarter. This process continues to generate sequences and corresponding forecast targets.

Table 4.6 1st Input and Output Sequence used in the LSTM model

Time period	Inflation Rate	Interest Rate	Real GDP	Potential Output	Inflation Forecast	Real GDP Forecast
2002-Q4	0.020805173	0.031277482	5.303673772	5.31523095	26.3289945	-0.068136509
2003-Q1	0.010289461	0.031896287	5.305750003	5.321096329	25.55277476	1.179269547
2003-Q2	-0.00193909	0.027838701	5.311009293	5.327286465	22.21899049	11.90999681
2003-Q3	-0.01369958	0.019080898	5.326266777	5.333757544	13.60194217	12.91669264
2003-Q4	-0.00451102	0.010335811	5.332461181			

Table 4.7 2nd Input and Output Sequence used in the LSTM model

Time period	Inflation Rate	Interest Rate	Real GDP	Potential Output	Inflation Forecast	Real GDP Forecast
2003-Q1	0.010289461	0.031896287	5.305750003	5.321096329	25.55277476	1.179269547
2003-Q2	-0.00193909	0.027838701	5.311009293	5.327286465	22.21899049	11.90999681
2003-Q3	-0.01369958	0.019080898	5.326266777	5.333757544	13.60194217	12.91669264
2003-Q4	-0.00451102	0.010335811	5.332461181	5.340455576	9.480111702	12.00637393
2004-Q1	-0.00810775	0.007462984	5.349563237			

Table 4.8 1st Input and Output Sequence used in the PINN model

Time period	Inflation Rate	Interest Rate	Real GDP	Potential Output	Inflation Forecast	Real GDP Forecast
2002-Q4	0.020805173	0.031277482	5.303673772	5.31523095	26.3289945	-0.068136509
2003-Q1	0.010289461	0.031896287	5.305750003	5.321096329	25.55277476	1.179269547
2003-Q2	-0.00193909	0.027838701	5.311009293	5.327286465	22.21899049	11.90999681
2003-Q3	-0.01369958	0.019080898	5.326266777	5.333757544	13.60194217	12.91669264
2003-Q4	-0.00451102	0.010335811	5.332461181	5.340455576	9.480111702	12.00637393

Table 4.9 2nd Input and Output Sequence used in the PINN model

Time period	Inflation Rate	Interest Rate	Real GDP	Potential Output	Inflation Forecast	Real GDP Forecast
2003-Q1	0.010289461	0.031896287	5.305750003	5.321096329	25.55277476	1.179269547
2003-Q2	-0.00193909	0.027838701	5.311009293	5.327286465	22.21899049	11.90999681
2003-Q3	-0.01369958	0.019080898	5.326266777	5.333757544	13.60194217	12.91669264
2003-Q4	-0.00451102	0.010335811	5.332461181	5.340455576	9.480111702	12.00637393
2004-Q1	-0.00810775	0.007462984	5.349563237	5.347321891	7.378891201	9.983548638

Table 4.6 and Table 4.7 show the 1st and 2nd input and output sequence used in the LSTM model. The green space gives the input sequence, the orange space gives the output sequence. Similarly, Table 4.8 and Table 4.9 show the 1st and 2nd input and output sequence utilized in the LSTM model. The green space gives the input sequence, the orange space gives the output sequence, the purple space represents exogeneous variables. In both LSTM model and PINN model, 6 inputs which are past values of inflation rate, interest rate, real GDP, potential output, inflation forecast, and real GDP forecast are used.

Splitting the dataset into two subsets is crucial for ensuring that the model can generalize effectively to new, unseen data. The training set is designed to teach the model the relationships between input data features and output data targets. During this stage, the model adjusts its parameters, such as weights in neural networks. The validation set impartially evaluates the model's fit on the training dataset while fine-tuning the hyperparameters of the model. Finally, the test set assesses the model's performance after training and validation. This study divides the macroeconomic time series data into 80% for training, 10% for validation, and 10% for testing.

4.2 Dataset for the New Keynesian Model for Mexico

The New Keynesian model is also employed on the Mexican economic series to apply the PIM-LSTM method. The real gross domestic product (GDP), the consumer price index (CPI), the short-term interest rate, the inflation forecast, and the real GDP forecast are used. We utilize a quarterly dataset of these variables spanning from 1998Q1 to 2024Q2 for Mexico. The real GDP and the CPI are sourced from the Federal Reserve Economic Data (FRED) at https://fred.stlouisfed.org/, while the short-term interest rate, the inflation forecast, and the real GDP forecast are retrieved from OECD Data Explorer.

Before conducting exploratory data analysis, some transformations on the dataset are applied. First, following Pfeifer's recommendations [33], some calculations are performed, shown in Equation (36). Secondly, the real GDP is seasonally adjusted by using Census X-13 procedure since the real GDP shows seasonal pattern. Then, two-sided Hodrick-Prescott (HP) filter is utilized to extract the trend component of seasonally adjusted real GDP. The trend component of seasonally adjusted real GDP represents the potential output. The potential output data obtained from HP filter is added to the dataset. Moreover, potential output, inflation forecasts, and GDP forecasts are considered exogenous variables.

4.2.1 Exploratory Data Analysis

Table 4.10 provides descriptive statistics of key macroeconomic indicators for Mexico, specifically the CPI inflation rate, interest rate, real GDP, potential output, inflation forecast, and GDP forecast. The CPI inflation rate has a mean close to zero (-0.0023) with a range from -0.0102 to 0.0173. Half of the CPI inflation rate values fall between -0.0052 and -0.0005. The interest rate has a mean of -0.0028. Half of the interest rate values are between -0.0066 and 0.0022. Similarly, the real GDP and the potential output values are closely aligned, with means of 6.7145 and 6.7140, respectively. The CPI inflation rate, the interest rate, the real GDP, the potential output are probably symmetrically distributed since their means and median values are close. The inflation forecast and the GDP forecast have high standard deviations (3.4654 and 4.1644). In contrast, the interest rate, the CPI inflation rate, the real GDP and the potential output show lower variability, with relatively small standard deviations indicating more stability.

Table 4.10 Descriptive Statistics of Mexico's Macroeconomic Series

	Inflation Rate	Interest Rate	Real GDP	Potential Output	Inflation Forecast	GDP Forecast
Minimum	−0.0102	−0.0083	6.6122	6.6005	2.2662	−20.2938
Q_1	−0.0052	−0.0066	6.6663	6.6705	3.7983	0.9048
Median	−0.0022	−0.0037	6.7101	6.7141	4.5085	2.3488
Mean	−0.0023	−0.0028	6.7145	6.7140	5.6325	1.9443
Standard Dev.	0.0045	0.0057	0.0550	0.0545	3.4654	4.1644
Q_3	−0.0005	−0.0022	6.7652	6.7684	5.9476	3.5132
Maximum	0.0173	0.0236	6.8018	6.7985	18.6328	22.7001

There is no missing data in any of the series. The time series graphs of Mexico's macroeconomic series are shown below. Figure 4.7 illustrates the changes in the CPI inflation rate from 1998Q1 to 2024Q2. The time series plot of the CPI inflation rate exhibits irregular cycle pattern over time. Until the last quarter of 2007, a downward

trend was observed. After that, small fluctuations around zero occurred. Figure 4.8 shows how the interest rate changed from 1998Q1 to 2024Q2. The interest rate shows no seasonal pattern and follows a decreasing trend. Figure 4.9 illustrates the changes in real GDP from 1998Q1 to 2024Q2. The real GDP shows no seasonal behavior and demonstrates an overall upward trend during this period, despite occasional short-term declines.

By examining the sACF and the sPACF in Figures 4.10, 4.11, and 4.12, inflation rate and interest rate could be stationary since their ACFs show exponential decay, their PACFs cut off after lags 1 and 2. However, the ACF of the real GDP shows slow decay, indicating that it is non-stationary.

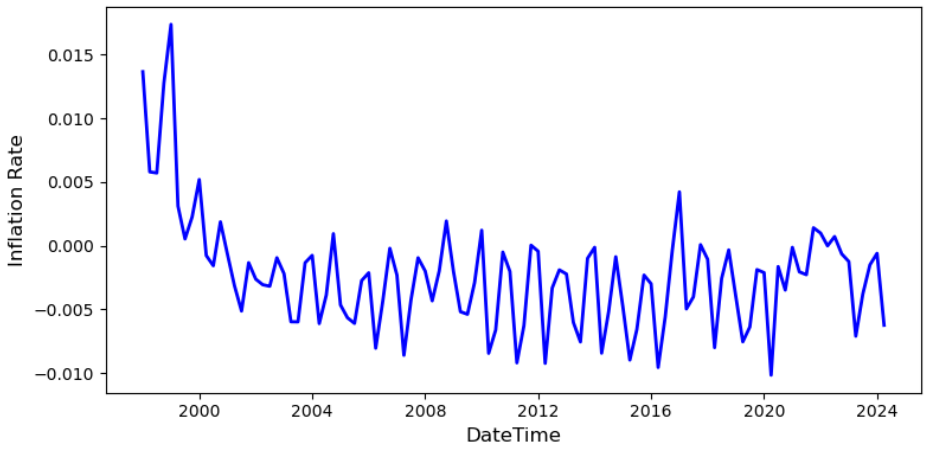

Figure 4.7 The Time Series Plot of Mexico's Inflation Rate

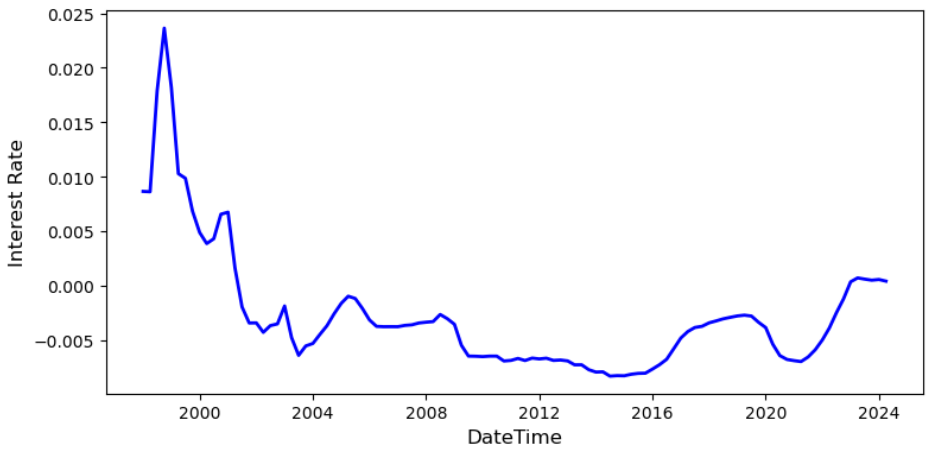

Figure 4.8 The Time Series Plot of Mexico's Interest Rate

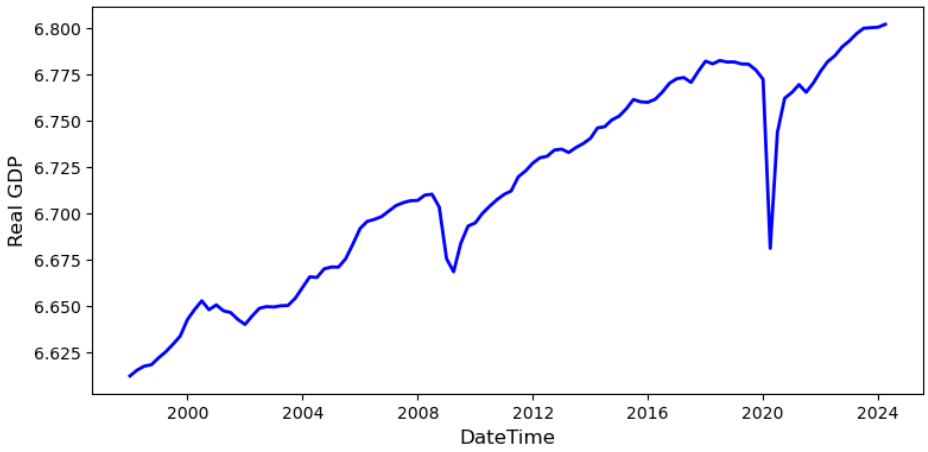

Figure 4.9 The Time Series Plot of Mexico's Real GDP

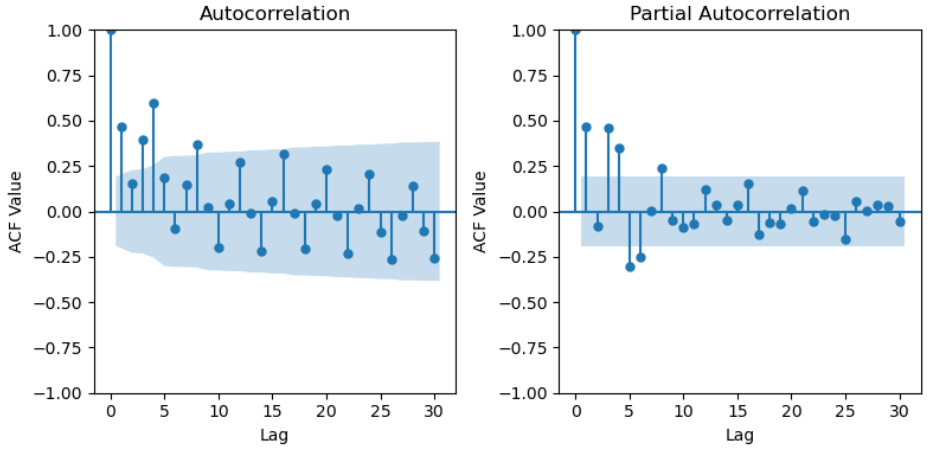

Figure 4.10 The sACF and sPACF of Mexico's Inflation Rate

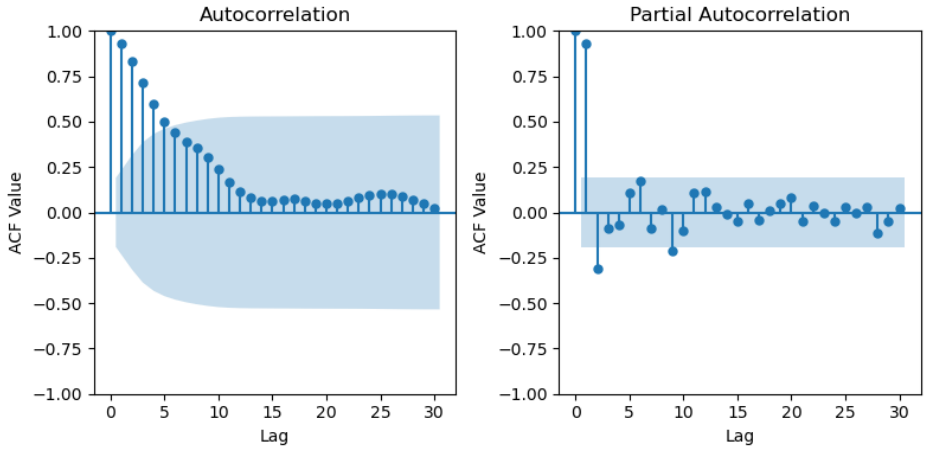

Figure 4.11 The sACF and sPACF of Mexico's Interest Rate

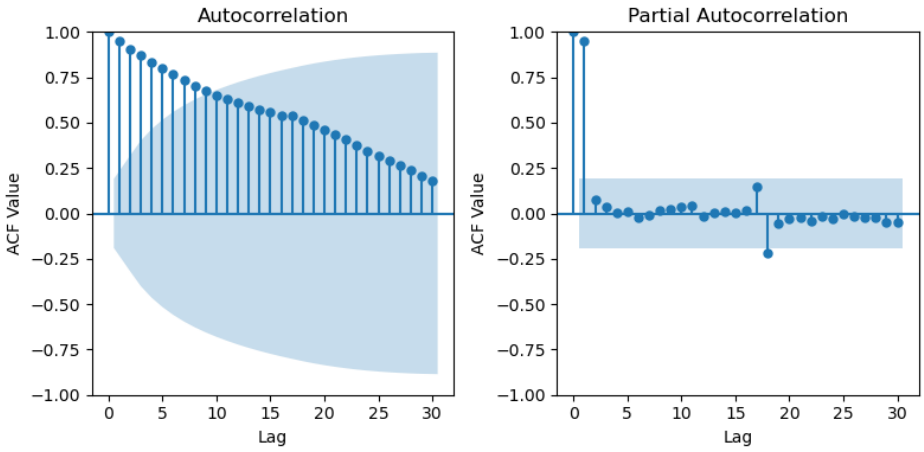

Figure 4.12 The sACF and sPACF of Mexico's Real GDP

Augmented Dickey-Fuller (ADF) Test

ADF test results show that the CPI inflation rate and the interest rate do not have unit roots, so these series are stationary. However, the real GDP has unit roots, indicating nonstationarity.

Table 4.11 ADF test for Inflation Rate

Test Statistic	−4.1983
Number of Lags Used	11
P-value	0.0007

Table 4.12 ADF test for Interest Rate

Test Statistic	−2.9846
Number of Lags Used	11
P-value	0.0364

Table 4.13 ADF test for Real GDP

Test Statistic	-1.0587
Number of Lags Used	2
P-value	0.7312

Toda-Yamamoto Causality Test

The Toda-Yamamoto causality test results presented in Table 4.14 reveal several significant relationships among the variables. Specifically, the interest rates and the real GDP Granger cause the inflation rate. The real GDP Granger causes the interest rate. These results suggest the presence of causal relationships among certain variables, indicating that multivariate time series analysis is applicable to this dataset.

Table 4.14 Toda-Yamamoto Causality Test Result

X => Y			P value
Interest Rate	=>	Inflation Rate	0.026
Interest Rate	=>	Real GDP	0.984
Inflation Rate	=>	Interest Rate	0.477
Inflation Rate	=>	Real GDP	0.726
Real GDP	=>	Interest Rate	0.047
Real GDP	=>	Inflation Rate	0.024

4.2.2 Data Preprocessing

Min-Max normalization is applied to scale the data while preserving its overall distribution. The data is transformed into sequences of inputs and outputs, allowing the model to learn from past input values and predict output values.

A sliding window approach is implemented with a stride of one quarter to create these input sequences. Each output sequence consists of three values, each representing a one-step-ahead forecast based on previous values of the series. This study splits the time series data into 80% for training, 10% for validation, and 10% for testing.

4.3 Dataset for Dividend-Augmented Goodwin-Keen Model

The Dividend-Augmented Goodwin-Keen model (DAGKM) for USA is also employed to apply the PIM-LSTM method. The model is composed of several differential equations for the wage share, the employment rate, and debt ratio. By the methodology suggested by Bailly et al. [3], our research employs a framework where the Goodwin-Keen model characterizes an economy comprised solely of households and private firms, intentionally omitting taxes and public investments. Moreover, financial activities affect the dynamics of the model only through loan provisions and interest payments by the non-financial sector. Employment, investment, and value-added are not directly impacted by banking activities in the model, leading to the exclusion of the banking sector from the analyzed time series. As a result, the data series are designed to closely represent the nonfinancial private sector, with most empirical variables defined as the combination of corporate and non-corporate nonfinancial business data. The series were primarily gathered from the FRED at https://fred.stlouisfed.org/. The series were collected quarterly over the period from 1959 to 2019. Additional information regarding the datasets used can be found in the Appendix (Table A.1).

The wage share at time t, ω_t^o, is defined as the ratio of real compensation of employees in the non-financial private sector to total real output:

$$\omega_t^o = \frac{Compensation\ of\ Employees_t^{nonfi}}{Y_t^o} \tag{39}$$

where $Y_t^o = Gross\ Value\ A._t^{nonfi} - Cons.of\ Fixed\ Capital_t^{nonfi} - Net\ Taxes\ on\ Prod_t^{nonfi}$.

The employment rate at time t, λ_t^o, is calculated from employment and unemployment series:

$$\lambda_t^o = \frac{Employment_t^{private\ nonagri}}{Employment_t^{private\ nonagri} + Unemployment_t^{private\ nonagri}} \tag{40}$$

As the employment rate exhibits seasonal behavior, seasonality is removed using the Seasonal Trend Decomposition using LOESS procedure [10]. The real debt ratio, d_t^o, is described as the ratio of the net debt level to total real output:

$$d_t^o = \frac{D_t^o}{Y_t^o} \tag{41}$$

where $D_t^o = Debt\ Security_t^{corpo} + Loans_t^{nonfi} - Saving\ Deposit_t^{nonfi}$.

4.3.1 Exploratory Data Analysis

Table 4.15 presents summary statistics for key economic indicators in the USA. The wage share ranges from 0.6269 to 0.7240, with an average of 0.6770 and a small standard deviation of 0.0214, indicating small variation. In contrast, the debt ratio has a broader range from 0.5563 to 1.5954, with a higher standard deviation of 0.2690, reflecting greater variability around its average of 1.0655. The employment rate falls between 0.8893 and 0.9562, averaging 0.9399, and has a standard deviation of 0.0168, suggesting it is relatively stable. Half of the wage share values are between 0.6615 and 0.6930. Half of the debt ratio values are between 0.8592 and 1.2633. Similarly, half of the employment rate values are between 0.9284 and 0.9510. Notably, the mean and median for each series are almost equal, indicating that all series are symmetrically distributed. There is no missing value in the series.

Table 4.15 Descriptive Statistics of US Economic Indicators

	Wage Share	**Debt Ratio**	**Employment Rate**
Minimum	0.6269	0.5563	0.8893
Q_1	0.6615	0.8592	0.9284
Median	0.6799	1.0858	0.9411
Mean	0.6770	1.0655	0.9390
Standard Dev.	0.0214	0.2690	0.0168
Q_3	0.6930	1.2633	0.9510
Maximum	0.7240	1.5954	0.9669

Figure 4.13 shows how the wage share in the US over time. The wage share rises significantly from the 1960s to 1980, peaking above 0.72. After 1980, it fluctuates and generally decreases until 2010s. Figure 4.14 shows the change in the US debt ratio over time. The debt ratio increased steadily over time. The debt ratio increased steadily from around 0.6 in 1959 to above 1.4 in 2020. There are noticeable short-term fluctuations but a sharp decline in 2010. After this decline, the debt ratio continues to rise. Figure 4.15 depicts the change in the US employment rate over time. The US employment rate displays an irregular cyclical pattern. Overall, the employment rate fluctuates between a low of 0.89 and a high of 0.97. Figures 4.16, 4.17, and 4.18 indicates that all sACF plots show slow decay, indicating that these three series are not stationary.

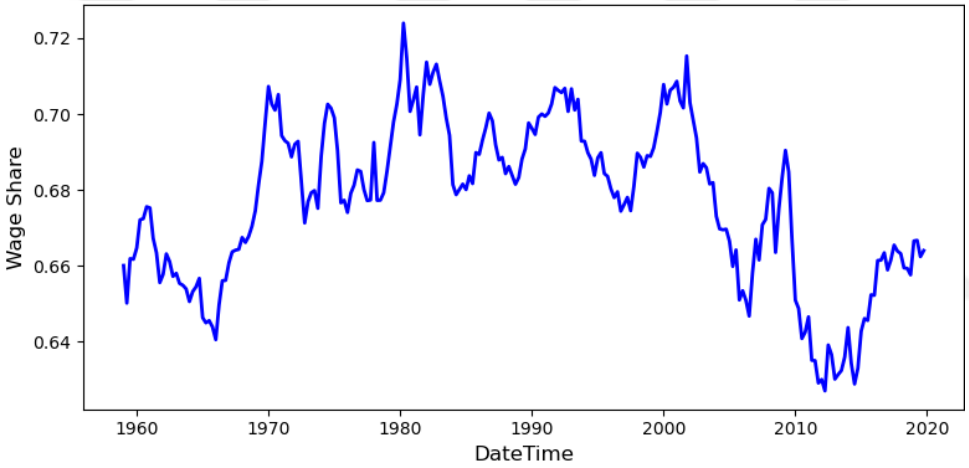

Figure 4.13 The Time Series Plot of the Wage Share of US

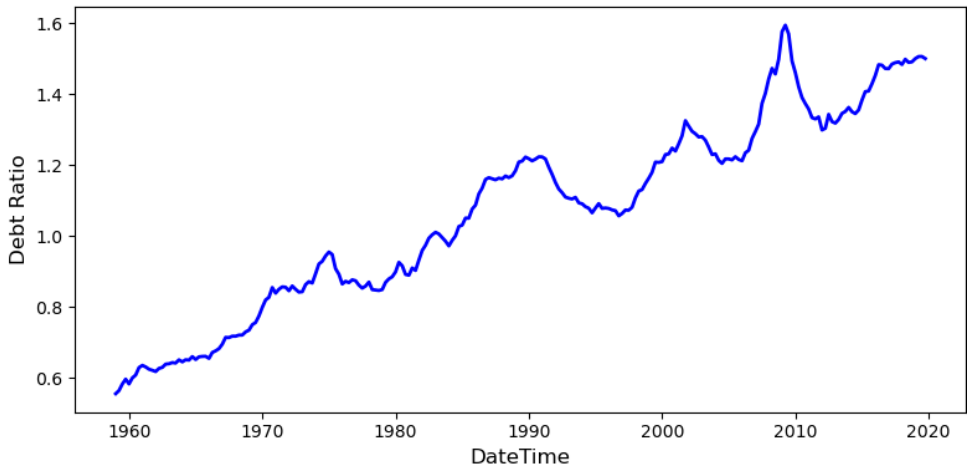

Figure 4.14 The Time Series Plot of the Debt Ratio of US

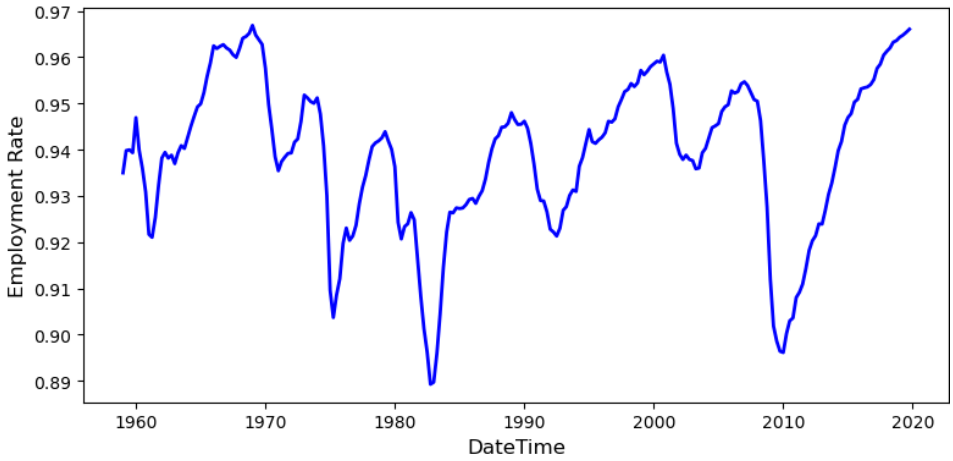

Figure 4.15 The Time Series Plot of the Employment Rate of US

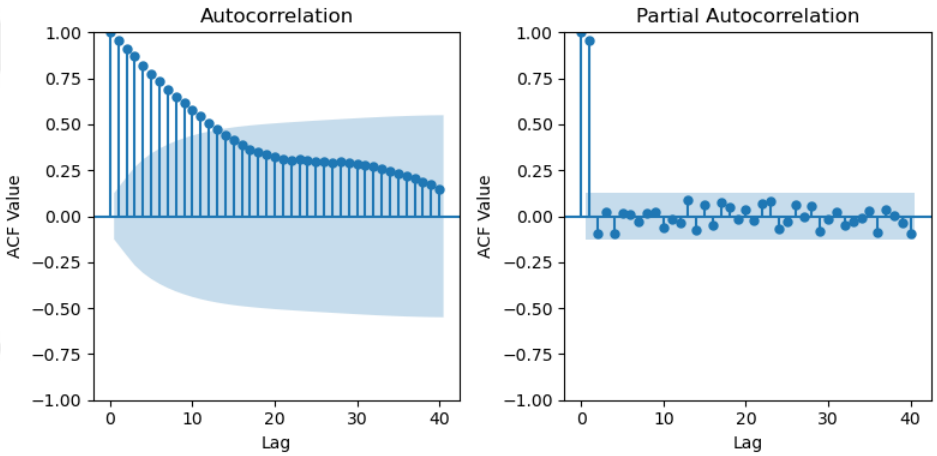

Figure 4.16 The sACF and sPACF of Wage Share of US

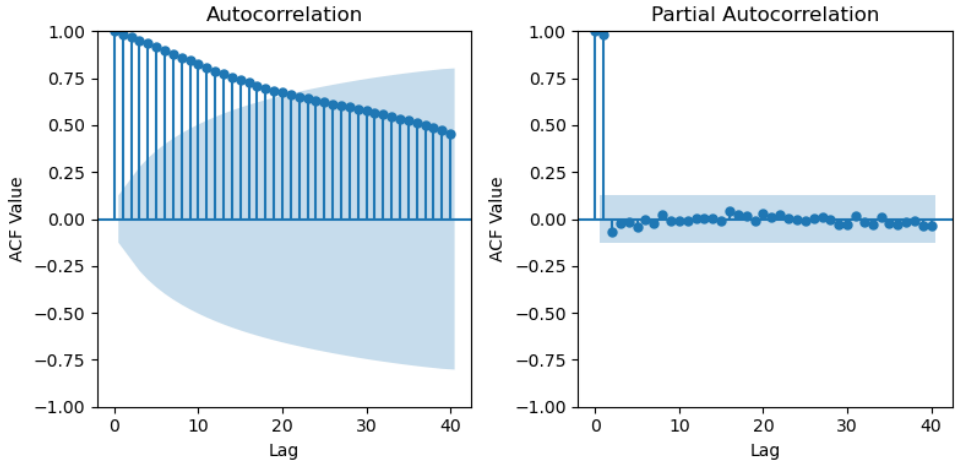

Figure 4.17 The sACF and sPACF of Debt Ratio of US

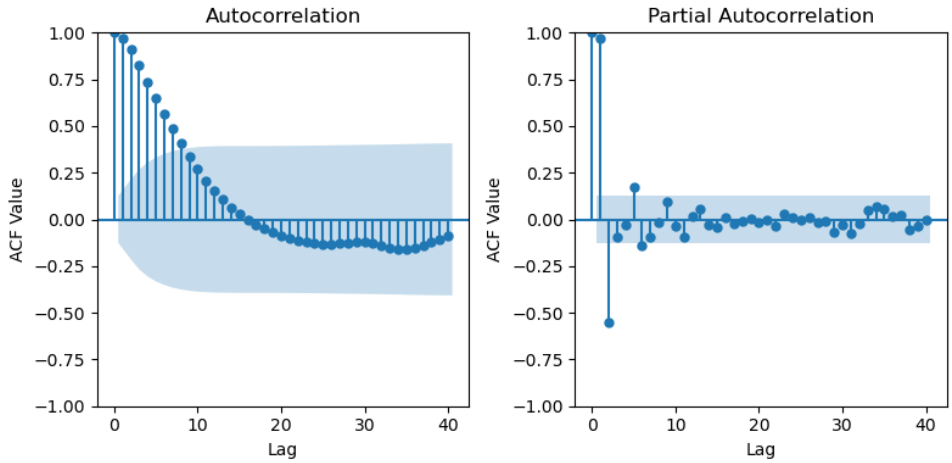

Figure 4.18 The sACF and sPACF of Employment Rate of US

Augmented Dickey-Fuller (ADF) Test

ADF test results show that the wage share, debt ratio, and employment rate exhibit unit roots, indicating nonstationarity.

Table 4.16 ADF test for US Wage Share

Test Statistic	−2.204730
Number of Lags Used	0
P-value	0.204533

Table 4.17 ADF test for US Debt Ratio

Test Statistic	−1.075899
Number of Lags Used	1
P-value	0.724587

Table 4.18 ADF test for US Employment Rate

Test Statistic	−2.789385
Number of Lags Used	13
P-value	0.059806

Toda-Yamamoto Causality Test

The Toda-Yamamoto causality test results presented in Table 4.19 reveal several significant relationships among the variables. Specifically, the debt ratio and employment rate are found to Granger-cause the wage share. Additionally, the employment rate also Granger-causes the debt ratio. However, the wage share does not Granger-cause either the debt ratio or the employment rate. These results suggest the presence of causal relationships among certain variables, indicating that multivariate time series analysis is applicable to this dataset.

Table 4.19 Toda-Yamamoto Causality Test Result

X => Y			**P value**
Wage Share	=>	Debt Ratio	0.056
Debt Ratio	=>	Wage Share	0.030
Wage Share	=>	Employment Rate	0.189
Employment Rate	=>	Wage Share	0.015
Debt Ratio	=>	Employment Rate	0.301
Employment Rate	=>	Debt Ratio	0.000

4.3.2 Data Preprocessing

Min-Max normalization scales the data to a fixed range, typically between 0 and 1, while preserving its overall distribution. The data is transformed into sequences of inputs and outputs, allowing the model to learn from past input values and predict output values. To create these input sequences, a sliding window approach is implemented with a stride of one quarter, which indicates the step size used to shift the window and generate new input-output pairs. Each output sequence consists of three values, each representing a one-step-ahead forecast based on previous values of the series. This study splits the time series data into 80% for training, 10% for validation, and 10% for testing.

4.4 Experiment Setup

In this study, deep learning models for forecasting time series, as well as data processing and analysis, are developed using the Python programming language. Its recognized flexibility and dependability in building these models, especially in the realm of machine learning, is well-established. Moreover, Python has many libraries and packages created for scientific computing and data analysis. This offers exciting opportunities for users.

PyTorch, an open-source deep learning framework, is used for this study due to its ability to facilitate flexible model execution and debugging using dynamic graphs. In addition, its automatic differentiation engine simplifies the computation of gradients in deep learning applications. Furthermore, PyTorch's support for CUDA enables efficient execution of computations on graphical processing units (GPUs). Ray Tune is preferred for performing hyperparameter tuning on deep learning models. Ray Tune helps find the best combination of parameters in deep learning models. It is a widely used tool for distributed hyperparameter tuning in the industry, with the most up-to-date hyperparameter search algorithms. Ray Tune also works seamlessly with TensorBoard and other analysis libraries, and it facilitates distributed training through Ray's distributed machine learning engine.

The PIM-LSTM model is a deep learning architecture built using PyTorch, designed to process sequential data with a Long Short-Term Memory (LSTM) network. It operates on CUDA for efficient GPU acceleration. The model consists of multiple configurable layers, including an LSTM layer, linear layers, layer normalization, dropout regularization, and residual connections. The LSTM layer processes input sequences with a specified number of hidden units and layers while applying dropout if more than one layer is used. Afterward, the output undergoes layer normalization and dropout to stabilize training and prevent overfitting. The architecture is dynamic, allowing different numbers of fully connected layers based on a parameter that defines the number of linear layers. If no additional layers are specified, the LSTM output is directly mapped to the final predictions. When one or more linear layers

are included, they are followed by ReLU activations, layer normalization, and residual connections to enhance gradient flow and model stability. In the forward pass, the LSTM processes the input sequence, followed by conditional linear transformations and residual connections, ultimately producing a final prediction.

In this experiment, we conduct hyperparameter tuning with Ray Tune to find the best hyperparameters such as number of hidden units, the number of linear layers, the number of LSTM layer etc. We determined the appropriate hyperparameters by setting the maximum epoch in Ray Tune to 100 and the hyperparameter trial epoch to 100 and using 50 or 75 different hyperparameter combination trials. Ray Tune is a powerful library for hyperparameter tuning, while Optuna is a flexible and efficient framework for hyperparameter optimization. When used together with OptunaSearch, they create a scalable, distributed, and effective solution for hyperparameter tuning. Therefore, for our hyperparameter tuning needs, we choose to use Ray Tune alongside OptunaSearch to explore various model configurations. The search space includes LSTM architecture parameters (hidden sizes, number of layers, dropout probabilities), learning rates, and batch sizes. The tuning process optimizes the model based on validation loss using Asynchronous Successive Halving (ASHA) for efficient resource allocation. The ASHA Scheduler is a useful tool for managing resources and stopping tasks early in hyperparameter optimization with Ray Tune. It helps with large experiments by quickly removing trials that are not performing well and reallocating resources to better candidates. ASHA monitors ongoing trials and halts those that fail to meet performance standards, reallocating resources to more promising configurations. Unlike traditional methods, it operates asynchronously and terminates trials without waiting for all to finish.

In the training process in our proposed model, the used loss function combines Mean Squared Error (MSE) with a cross-correlation loss to align model predictions with real-world dependencies. Additionally, prior knowledge constraints are incorporated into the training process by enforcing relationships derived from economic models. Additionally, gradient clipping is applied to prevent exploding gradients, and learning rate scheduler "*ReduceLROnPlateau*" is applied. It dynamically reduces the

learning rate when a monitored metric which is validation loss stops improving. Instead of reducing the learning rate at fixed intervals, this scheduler adjusts it only when necessary, helping the model continue learning efficiently. The model's performance is periodically assessed using a validation set, and checkpointing is employed to save model progress and resume training if interrupted. After performing hyperparameter tuning in Ray Tune, the best training parameters are selected based on the lowest validation loss. Using the best-trained model, we then generate the forecast values for the test set.

While constructing the PIM-LSTM model, a prior knowledge-based approach was adopted to define the prior loss function. This prior knowledge consists of specific mathematical equations, within which certain parameters exist. These parameters have either been previously calibrated or estimated. In this study, two different strategies were employed. Firstly, the estimated or calibrated parameter values were directly incorporated into the model (referred to PIM-LSTM1). Secondly, these parameters were assumed to be unknown and were introduced into the model as hyperparameters (referred to PIM-LSTM2). This approach allowed the model to learn these parameters independently, facilitating a data-driven optimization process. By combining both strategies, the model's flexibility was enhanced, enabling the development of a learning mechanism that leverages both prior knowledge and data-driven learning.

4.4.1 Hyperparameter Tuning for New Keynesian Model for Turkiye

Hyperparameter and model parameter tuning are conducted for the PIM-LSTM architecture for Turkiye's New Keynesian Model. Firstly, for the PIM-LSTM model, the calibrated parameters specific to the New Keynesian model are applied. These calibrated parameters for Turkiye are sourced from the article by Güloğlu and Güngör [12]. Table 4.20 shows the calibrated parameters for the New Keynesian model related to Turkiye.

Table 4.20 Calibrated Parameters from Güloğlu's Article [12]

$\alpha = 0.33$	Capital share in Production	$\sigma = 1$	Relative Risk Aversion
$\beta = 0.99$	Discount Factor	$\theta = 0.6699$	Calvo Parameter
$\in = 6$	Elasticity of Substitution	$\phi_\pi = 1.9959$	Feedback parameter of inflation
$\varphi = 1$	Frisch Elasticity	$\phi_y = 0.5122$	Feedback parameter of output gap

Table 4.21 shows the detailed setup for PIM-LSTM model with known parameters. The hyperparameter tuning experiment for PIM-LSTM model with known parameters in the New Keynesian Model was conducted using Ray Tune with OptunaSearch and an Async Hyperband Scheduler, running 75 trials to minimize validation loss. The best model configuration included 32 hidden units in the first LSTM layer, one LSTM layer, and one linear layer, with a dropout rate of 0.2. The best training parameters were found using the AdamW optimizer, Xavier weight initialization, and Normal bias initialization, with batch shuffling enabled. The optimal batch size was 4 and the input sequence length was 4. The learning rate was 0.0056171, with a weight decay of 0.002717, and learning rate scheduling was applied using Reduce Learning Rate on Plateau, with a patience of 2 and a reduction factor of 0.8. The training process was set to run for a maximum of 100 epochs, with a grace period of 10 and a reduction factor of 2. $\alpha_1 = 0.1$ and $\alpha_2 = 0.1$ determine how much influence each loss component has on the total loss function: $\mathcal{L}_{total} = MSE_{train} + \alpha_1 MSE_{prior} + \alpha_2 \mathcal{L}_{CCF}$.

Secondly, the parameters in the New Keynesian model were treated as unknowns and integrated into the model as hyperparameters when implementing the PIM-LSTM model for the New Keynesian model. Table 4.22 indicates the detailed setup for PIM-LSTM with unknown parameters. The hyperparameter tuning experiment for the PIM-LSTM model with unknown parameters in the New Keynesian Model utilized Ray Tune with OptunaSearch and an Async Hyperband Scheduler, conducting 75 trials to minimize validation loss. The best model configuration selected 32 hidden units in the first LSTM layer, one LSTM layer, two linear layers,

and a dropout rate of 0.2. The best training parameters included the AdamW optimizer, Xavier weight initialization, and Normal bias initialization, with batch shuffling enabled. The optimal batch size was 4, input sequence length was 16, and learning rate was 0.001736, with weight decay of 0.000005. Learning rate scheduling was applied using Reduce Learning Rate on Plateau, with a patience of 2 and a reduction factor of 0.8. The tuning also optimized New Keynesian model parameters ($\beta, \theta, \alpha, \in, \sigma,\ \varphi\ ,\ \phi_\pi$ and ϕ_y), with the best values found at $\beta = 0.36058$, $\theta = 0.75432$, $\alpha = 0.66505$, $\in = 7$, $\sigma = 5$, $\varphi = 4$, $\phi_\pi = 1.8387$ and $\phi_y = 0.5644$. $\alpha_1 = 0.1$ and $\alpha_2 = 0.1$ are weighting coefficients.

Table 4.21 Hyperparameter tuning experiment setup for PIM-LSTM1 Model for Turkiye's New Keynesian Model

Hyperparameter Tuning	**Trials**	**Best Model Parameters**
Tuning Environment	Ray Tune	
Search Algorithm	OptunaSearch	
Scheduler	AsyncHyperBandScheduler	
Hyperparameter Trials	75	
Tuning Objective	Minimize Validation Loss	
Hyperparameter Trial Epochs	100	
LSTM Layer 1 Hidden Size	16, 32, 64, 128, 256,512	**32**
LSTM Layer Num Layer	1, 2, 3, 4, 5	**1**
Linear Layer Num Layer	0, 1, 2	**1**
Dropout 1	0.1,0.2, 0.3, 0.4, 0.5	**0.2**
Linear Layer 1 Hidden Size	16, 32, 64, 128, 256,512	**32**
Linear Layer 2 Hidden Size	16, 32, 64, 128, 256,512	**256**
Model Training Parameter Tuning	**Trials**	**Best Training Parameters**
Optimizers	Adam, RMSProp, SGD, AdamW	**AdamW**
Weight Initialization	Xavier, Kaiming, Normal	**Xavier**
Bias Initialization	Zeros, Ones, Normal	**Normal**
Batch Shuffle	On	**On**
Batch Size	4, 8, 16, 32,64	**4**
Input Sequence Length	4, 8, 12, 16	**4**
α_1	0.1, 0.3, 0.5, 0.7, 0.9	**0.1**
α_2	0.1, 0.3, 0.5, 0.7, 0.9	**0.1**
Learning Rate	Log Uniform: Min:0.00001 Max:0.01	**0.0056171**
Weight Decay	Log Uniform: Min:0.000001 Max:0.01	**0.002717**
Learning Rate Scheduler	Reduce Learning Rate on Plateau	
Learning Rate Scheduler Patience	2	
Learning Rate Scheduler Factor	0.8	
Best Model Training	**Additional Setting**	
Max Epoch	100	
Grace Period	10	
Reduction Factor	2	

Table 4.22 Hyperparameter tuning experiment setup for PIM-LSTM2 Model for Turkiye's New Keynesian Model

Hyperparameter Tuning	**Trials**	**Best Model Parameters**
Tuning Environment	Ray Tune	
Search Algorithm	OptunaSearch	
Scheduler	AsyncHyperBandScheduler	
Hyperparameter Trials	75	
Tuning Objective	Minimize Validation Loss	
Hyperparameter Trial Epochs	100	
LSTM Layer 1 Hidden Size	16, 32, 64, 128, 256,512	**32**
LSTM Layer Num Layer	1, 2, 3, 4, 5	**1**
Linear Layer Num Layer	0, 1, 2	**2**
Dropout 1	0.1,0.2, 0.3, 0.4, 0.5	**0.2**
Linear Layer 1 Hidden Size	16, 32, 64, 128, 256,512	**128**
Linear Layer 2 Hidden Size	16, 32, 64, 128, 256,512	**32**
Model Training Parameter Tuning	**Trials**	**Best Training Parameters**
Optimizers	Adam, RMSProp, SGD, AdamW	**AdamW**
Weight Initialization	Xavier, Kaiming, Normal	**Xavier**
Bias Initialization	Zeros, Ones, Normal	**Normal**
Batch Shuffle	On	**On**
Batch Size	4, 8, 16, 32,64	**4**
Input Sequence Length	4, 8, 12, 16	**16**
α_1	0.1, 0.3, 0.5, 0.7, 0.9	**0.1**
α_2	0.1, 0.3, 0.5, 0.7, 0.9	**0.1**
Learning Rate	Log Uniform: Min:0.00001 Max:0.01	**0.001736**
Weight Decay	Log Uniform: Min:0.000001 Max:0.01	**0.000005**
Learning Rate Scheduler	Reduce Learning Rate on Plateau	
Learning Rate Scheduler Patience	2	
Learning Rate Scheduler Factor	0.8	
β	Uniform Min:0.01 Max:0.999	**0.36058**
θ	Uniform Min:0.6 Max:0.9	**0.75432**
α	Uniform Min:0.01 Max:0.999	**0.66505**
$\in$	1,3,5,7,9,11	**7**
σ	1,3,5,7,9,11	**5**
φ	1,2,3,4	**4**
ϕ_π	Uniform Min:1 Max:2	**1.83874**
ϕ_y	Uniform Min:0.01 Max:0.999	**0.56441**
Best Model Training	**Additional Setting**	
Max Epoch	100	
Grace Period	10	
Reduction Factor	2	

4.4.2 Hyperparameter Tuning for New Keynesian Model for Mexico

Hyperparameter and model parameter tuning are performed for the PIM-LSTM architecture for Mexico's New Keynesian Model. Initially, for the PIM-LSTM model, the calibrated parameters specific to Mexico's New Keynesian model are utilized. These calibrated parameters are sourced from the article by Zendejas-Fonseca et al. [50]. Table 4.23 displays the calibrated parameters for Mexico's New Keynesian model.

Table 4.23 Calibrated Parameters from Zendejas-Fonseca's Article [50]

$\alpha = 0.33$	Capital share in Production	$\sigma = 1$	Relative Risk Aversion
$\beta = 0.99$	Discount Factor	$\theta = 0.6699$	Calvo Parameter
$\in = 6$	Elasticity of Substitution	$\phi_\pi = 1.5$	Feedback parameter of inflation
$\varphi = 1$	Frisch Elasticity	$\phi_y = 0.12$	Feedback parameter of output gap

Table 4.24 shows the detailed setup for PIM-LSTM model with known parameters. The hyperparameter tuning experiment for PIM-LSTM model with known parameters in the New Keynesian Model was conducted using Ray Tune with OptunaSearch and an Async Hyperband Scheduler, running 75 trials to minimize validation loss. The best model configuration included 32 hidden units in the first LSTM layer, one LSTM layer, and no linear layer, with a dropout rate of 0.3. The best training parameters were found using the Adam optimizer, Kaiming weight initialization, and Ones bias initialization, with batch shuffling enabled. The optimal batch size was 8, and the input sequence length was 16. The learning rate was 0.007785, with a weight decay of 0.000019, and learning rate scheduling was applied using Reduce Learning Rate on Plateau, with a patience of 2 and a reduction factor of 0.8. The training process was set to run for a maximum of 100 epochs, with a grace period of 10 and a reduction factor of 2. $\alpha_1 = 0.1$ and $\alpha_2 = 0.1$ determine how much influence each loss component has on the total loss function: $\mathcal{L}_{total} = MSE_{train} + \alpha_1 MSE_{prior} + \alpha_2 \mathcal{L}_{CCF}$.

Secondly, the parameters in the New Keynesian model were treated as unknowns and integrated into the model as hyperparameters when implementing the PIM-LSTM model for the New Keynesian model. Table 4.25 indicates the detailed setup for PIM-LSTM with unknown parameters. The hyperparameter tuning experiment for the PIM-LSTM model with known parameters in the New Keynesian Model utilized Ray Tune with OptunaSearch and an Async Hyperband Scheduler, conducting 75 trials to minimize validation loss. The best model configuration selected 256 hidden units in the first LSTM layer, one LSTM layer, no linear layers, and a dropout rate of 0.2. The best training parameters included the Adam optimizer, Kaiming weight initialization, and Ones bias initialization, with batch shuffling enabled. The optimal batch size was 32, input sequence length was 4, and learning rate was 0.003166, with weight decay of 0.000257. Learning rate scheduling was applied using Reduce Learning Rate on Plateau, with a patience of 2 and a reduction factor of 0.8. The tuning also optimized New Keynesian model parameters ($\beta, \theta, \alpha, \in, \sigma, \varphi, \phi_\pi$ and ϕ_y), with the best values found at $\beta = 0.86045$, $\theta = 0.61734$, $\alpha = 0.75074$, $\in = 5$, $\sigma = 9$, $\varphi = 3$, $\phi_\pi = 1.76522$ and $\phi_y = 0.59629$. Also, the weighting coefficients are $\alpha_1 = 0.3$ and $\alpha_2 = 0.1$.

Table 4.24 Hyperparameter tuning experiment setup for PIM-LSTM1 Model for Mexico's New Keynesian Model

Hyperparameter Tuning	**Trials**	**Best Model Parameters**
Tuning Environment	Ray Tune	
Search Algorithm	OptunaSearch	
Scheduler	AsyncHyperBandScheduler	
Hyperparameter Trials	75	
Tuning Objective	Minimize Validation Loss	
Hyperparameter Trial Epochs	100	
LSTM Layer 1 Hidden Size	16, 32, 64, 128, 256,512	**32**
LSTM Layer Num Layer	1, 2, 3, 4, 5	**1**
Linear Layer Num Layer	0, 1, 2	**0**
Dropout 1	0.1,0.2, 0.3, 0.4, 0.5	**0.3**
Linear Layer 1 Hidden Size	16, 32, 64, 128, 256,512	**512**
Linear Layer 2 Hidden Size	16, 32, 64, 128, 256,512	**16**
Model Training Parameter Tuning	**Trials**	**Best Training Parameters**
Optimizers	Adam, RMSProp, SGD, AdamW	**Adam**
Weight Initialization	Xavier, Kaiming, Normal	**Kaiming**
Bias Initialization	Zeros, Ones, Normal	**Ones**
Batch Shuffle	On	**On**
Batch Size	4, 8, 16, 32, 64	**8**
Input Sequence Length	4, 8, 12, 16	**16**
α_1	0.1, 0.3, 0.5, 0.7, 0.9	**0.1**
α_2	0.1, 0.3, 0.5, 0.7, 0.9	**0.1**
Learning Rate	Log Uniform: Min:0.00001 Max:0.01	**0.007785**
Weight Decay	Log Uniform: Min:0.000001 Max:0.01	**0.000019**
Learning Rate Scheduler	Reduce Learning Rate on Plateau	
Learning Rate Scheduler Patience	2	
Learning Rate Scheduler Factor	0.8	
Best Model Training	**Additional Setting**	
Max Epoch	100	
Grace Period	10	
Reduction Factor	2	

Table 4.25 Hyperparameter tuning experiment setup for PIM-LSTM2 Model for Mexico's New Keynesian Model

Hyperparameter Tuning	**Trials**	**Best Model Parameters**
Tuning Environment	Ray Tune	
Search Algorithm	OptunaSearch	
Scheduler	AsyncHyperBandScheduler	
Hyperparameter Trials	75	
Tuning Objective	Minimize Validation Loss	
Hyperparameter Trial Epochs	100	
LSTM Layer 1 Hidden Size	16, 32, 64, 128, 256,512	**256**
LSTM Layer Num Layer	1, 2, 3, 4, 5	**1**
Linear Layer Num Layer	0, 1, 2	**0**
Dropout 1	0.1,0.2, 0.3, 0.4, 0.5	**0.1**
Linear Layer 1 Hidden Size	16, 32, 64, 128, 256,512	**512**
Linear Layer 2 Hidden Size	16, 32, 64, 128, 256,512	**512**
Model Training Parameter Tuning	**Trials**	**Best Training Parameters**
Optimizers	Adam, RMSProp, SGD, AdamW	**Adam**
Weight Initialization	Xavier, Kaiming, Normal	**Kaiming**
Bias Initialization	Zeros, Ones, Normal	**Ones**
Batch Shuffle	On	**On**
Batch Size	4, 8, 16, 32,64	**32**
Input Sequence Length	4, 8, 12, 16	**4**
α_1	0.1, 0.3, 0.5, 0.7, 0.9	**0.3**
α_2	0.1, 0.3, 0.5, 0.7, 0.9	**0.1**
Learning Rate	Log Uniform: Min:0.00001 Max:0.01	**0.003166**
Weight Decay	Log Uniform: Min:0.000001 Max:0.01	**0.000257**
Learning Rate Scheduler	Reduce Learning Rate on Plateau	
Learning Rate Scheduler Patience	2	
Learning Rate Scheduler Factor	0.8	
β	Uniform Min:0.01 Max:0.999	**0.86045**
θ	Uniform Min:0.6 Max:0.9	**0.61734**
α	Uniform Min:0.01 Max:0.999	**0.75074**
$\in$	1,3,5,7,9,11	**5**
σ	1,3,5,7,9,11	**9**
φ	1,2,3,4	**3**
ϕ_π	Uniform Min:1 Max:2	**1.76522**
ϕ_y	Uniform Min:0.01 Max:0.999	**0.59629**
Best Model Training	**Additional Setting**	
Max Epoch	100	
Grace Period	10	
Reduction Factor	2	

4.4.3 Hyperparameter Tuning for DAGKM Model

Hyperparameter and model parameter tuning are conducted for the PIM-LSTM architecture for the DAGKM model. When working with the PIM-LSTM model, we utilize estimated parameters. These estimated parameters for the DAGKM model in the USA are derived from the article by Bailly et al. [3]. Table 4.26 exhibits the estimated parameters used in the DAGKM model for the USA.

Table 4.26 Estimate of Parameters from Bailly's Article [3]

$\rho = 0.0000352$	$r = 0.0126$	$\delta = 0.0427$
$k_1 = 0.0584$	$k_2 = 4.03$	$\Delta = 0.469$

Table 4.27 shows the detailed setup for PIM-LSTM model with known parameters. The hyperparameter tuning experiment for PIM-LSTM model with known parameters in the DAGKM model was conducted using Ray Tune with OptunaSearch and an Async Hyperband Scheduler, running 50 trials to minimize validation loss. The best model configuration selected 64 hidden units in the first LSTM layer, one LSTM layer, zero linear layer, and a dropout rate of 0.1. The best training parameters were determined using the RMSProp optimizer, Normal weight initialization, and Normal bias initialization, with batch shuffling enabled. The optimal batch size was 8, and the input sequence length was 4. The learning rate was found to be 0.005097, with a weight decay of 0.00006. Learning rate scheduling was applied using Reduce Learning Rate on Plateau, with a patience of 2 and a reduction factor of 0.8. The training process was set to run for a maximum of 100 epochs, with a grace period of 10 and a reduction factor of 2. $\alpha_1 = 0.1$ and $\alpha_2 = 0.5$ determine how much influence each loss component has on the total loss function: $\mathcal{L}_{total} = MSE_{train} + \alpha_1 MSE_{prior} + \alpha_2 \mathcal{L}_{CCF}$.

Table 4.28 shows the detailed setup for PIM-LSTM model with unknown parameters. The hyperparameter tuning experiment for PIM-LSTM model with

unknown parameters in the DAGKM model was conducted using Ray Tune with OptunaSearch and an Async Hyperband Scheduler, running 50 trials to minimize validation loss. The best model configuration selected 32 hidden units in the first LSTM layer, one LSTM layer, no linear layers, and a dropout rate of 0.4. The best training parameters were found using the RMSProp optimizer, Normal weight initialization, and Normal bias initialization, with batch shuffling enabled. The optimal batch size was 16, and the input sequence length was 4. The learning rate was 0.00764, with a weight decay of 0.00006. Learning rate scheduling was applied using Reduce Learning Rate on Plateau, with a patience of 2 and a reduction factor of 0.8. Additional parameters were optimized, with the best values set for $\rho = 0.00004, k_1 = 0.01935, k_2 = 5.09022, r = 0.00947, \delta = 0.03806, \Delta = 0.85094$. Also, the best weighting coefficients are $\alpha_1 = 0.1$ and $\alpha_2 = 0.3$.

Table 4.27 Hyperparameter tuning experiment setup for PIM-LSTM1 Model for DAGKM Model

Hyperparameter Tuning	**Trials**	**Best Model Parameters**
Tuning Environment	Ray Tune	
Search Algorithm	OptunaSearch	
Scheduler	AsyncHyperBandScheduler	
Hyperparameter Trials	50	
Tuning Objective	Minimize Validation Loss	
Hyperparameter Trial Epochs	100	
LSTM Layer 1 Hidden Size	16, 32, 64, 128, 256,512	**64**
LSTM Layer Num Layer	1, 2, 3, 4, 5	**1**
Linear Layer Num Layer	0, 1, 2	**0**
Dropout 1	0.1,0.2, 0.3, 0.4, 0.5	**0.1**
Linear Layer 1 Hidden Size	16, 32, 64, 128, 256,512	**64**
Linear Layer 2 Hidden Size	16, 32, 64, 128, 256,512	**256**
Model Training Parameter Tuning	**Trials**	**Best Training Parameters**
Optimizers	Adam, RMSProp, SGD, AdamW	**RMSProp**
Weight Initialization	Xavier, Kaiming, Normal	**Normal**
Bias Initialization	Zeros, Ones, Normal	**Normal**
Batch Shuffle	On	**On**
Batch Size	4, 8, 16, 32,64	**8**
Input Sequence Length	4, 8, 12, 16	**4**
α_1	0.1,0.3,0.5,0.7	**0.1**
α_2	0.1,0.3,0.5,0.7	**0.5**
Learning Rate	Log Uniform: Min:0.00001 Max:0.01	**0.005907**
Weight Decay	Log Uniform: Min:0.000001 Max:0.01	**0.00006**
Learning Rate Scheduler	Reduce Learning Rate on Plateau	
Learning Rate Scheduler Patience	2	
Learning Rate Scheduler Factor	0.8	
Best Model Training	**Additional Setting**	
Max Epoch	100	
Grace Period	10	
Reduction Factor	2	

Table 4.28 Hyperparameter tuning experiment setup for PIM-LSTM2 Model for DAGKM Model

Hyperparameter Tuning	**Trials**	**Best Model Parameters**
Tuning Environment	Ray Tune	
Search Algorithm	OptunaSearch	
Scheduler	AsyncHyperBandScheduler	
Hyperparameter Trials	50	
Tuning Objective	Minimize Validation Loss	
Hyperparameter Trial Epochs	100	
LSTM Layer 1 Hidden Size	16, 32, 64, 128, 256, 512	**32**
LSTM Layer Num Layer	1, 2, 3, 4, 5	**1**
Linear Layer Num Layer	0, 1, 2	**0**
Dropout 1	0.1, 0.2, 0.3, 0.4, 0.5	**0.4**
Linear Layer 1 Hidden Size	16, 32, 64, 128, 256, 512	**256**
Linear Layer 2 Hidden Size	16, 32, 64, 128, 256, 512	**256**
Model Training Parameter Tuning	**Trials**	**Best Training Parameters**
Optimizers	Adam, RMSProp, SGD, AdamW	**RMSProp**
Weight Initialization	Xavier, Kaiming, Normal	**Normal**
Bias Initialization	Zeros, Ones, Normal	**Normal**
Batch Shuffle	On	**On**
Batch Size	4, 8, 16, 32,64	**16**
Input Sequence Length	4, 8, 12, 16	**4**
α_1	0.1,0.3,0.5,0.7	**0.1**
α_2	0.1,0.3,0.5,0.7	**0.3**
Learning Rate	Log Uniform: Min:0.00001 Max:0.01	**0.00764**
Weight Decay	Log Uniform: Min:0.000001 Max:0.01	
Learning Rate Scheduler	Reduce Learning Rate on Plateau	**0.00006**
Learning Rate Scheduler Patience	2	
Learning Rate Scheduler Factor	0.8	
ρ	Uniform Min:0.0000144 Max:0.000044	**0.00004**
k_1	Uniform Min:0.00279 Max:0.148	**0.01935**
k_2	Uniform Min:2.47 Max:10	**5.09022**
r	Uniform Min:-0.0501 Max:0.0721	**0.00947**
δ	Uniform Min:0 Max:0.0937	**0.03806**
Δ	Uniform Min:0.000036 Max:0.894	**0.85094**
Best Model Training	**Additional Setting**	
Max Epoch	100	
Grace Period	10	
Reduction Factor	2	

4.5 Results

Firstly, we conducted multivariate time series forecasting for the New Keynesian model for Turkiye. Six deep learning models were utilized to predict the values of the series in the New Keynesian model. These models are the LSTM model and LSTM model with CCF loss, PINN model with Data-Driven Solution, PINN model with Data-Driven Discovery, PIM-LSTM model with known parameters (referred to as PIM-LSTM1), and PIM-LSTM model with unknown parameters (referred to as PIM-LSTM2). After performing hyperparameter tuning on all models for three series, the performance metrics, MAE, RMSE and MASE were calculated. Furthermore, the Vector Error Correction Model (VECM) is implemented for multivariate time series forecasting. Table 4.29 shows the out-of-sample forecast performance of these models for each series. In terms of overall performance across all indicators, the PIM-LSTM1 achieves the lowest MAE (0.0073), RMSE (0.0132), and MASE (0.8013), indicating superior accuracy.

Table 4.29 Test Set Model Performance for Turkiye's New Keynesian Model

Model	**MAE**	**RMSE**	**MASE**
LSTM Model	0.0140	0.0197	1.5288
PINN Model (Data-Driven Solution)	0.0093	0.0138	1.0194
PINN Model (Data-Driven Discovery)	0.0173	0.0255	1.8814
LSTM_CCF Model	0.0101	0.0201	1.1062
PIM-LSTM1	**0.0073**	**0.0132**	**0.8013**
PIM-LSTM2	0.0116	0.0171	1.2696
VECM Model	0.0095	0.014	1.9027

Figures 4.19, 4.20 and 4.21 depict the time series forecast plot for the CPI inflation rate, the short-term interest rate and real GDP of Turkiye. In these plots, the legend of "PIM-LSTM2" represents the PIM-LSTM model with unknown parameters, while "PIM-LSTM1" represents the PIM-LSTM model with known parameters.

Additionally, "PINN1" represents the PINN model with data-driven solutions and "PINN2" refers to the PINN model with data-driven discovery.

Figure 4.19 shows the performance of different models in forecasting the CPI inflation rate over time, compared to the actual CPI inflation values, which are represented by the "Test Set" in orange. Each colored and dashed line corresponds to a different model: LSTM, LSTM_CCF, PINN1, PINN2, PIM-LSTM2, PIM-LSTM1, and VECM. Most models struggle to fully capture the upward trend in inflation rate seen in the test set in late 2020. Nevertheless, among the models, the LSTM (green) and PINN1(purple) model forecasts are slightly closer to actual inflation values. Figure 4.20 shows the forecast of interest rates obtained by various models, including the LSTM model, LSTM_CCF model, PINN1, PINN2, PIM-LSTM2, PIM-LSTM1, and VECM. Overall, the forecasts from the PIM-LSTM1(red) model, PINN1(purple) model, and the PIM-LSTM2(light blue) model are closer to the actual interest rate values. Figure 4.21 shows the predictions of the real GDP using the LSTM model, LSTM_CCF model, PINN1, PINN2, PIM-LSTM2, PIM-LSTM1, and VECM. Both the PIM-LSTM1 (red) model and VECM (brown) provide a more consistent and realistic forecast.

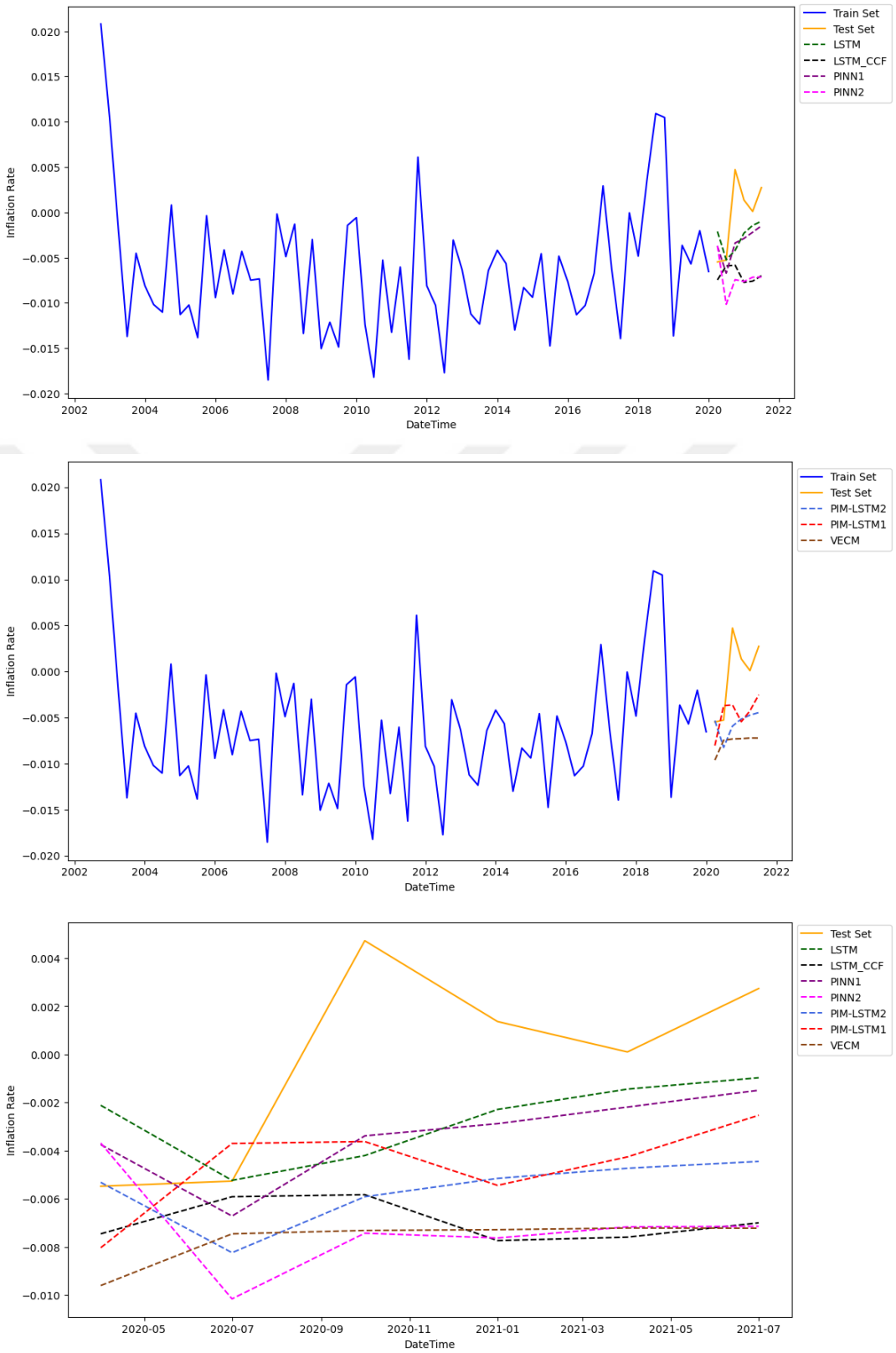

Figure 4.19 Forecast Plot of Turkiye's Inflation Rate

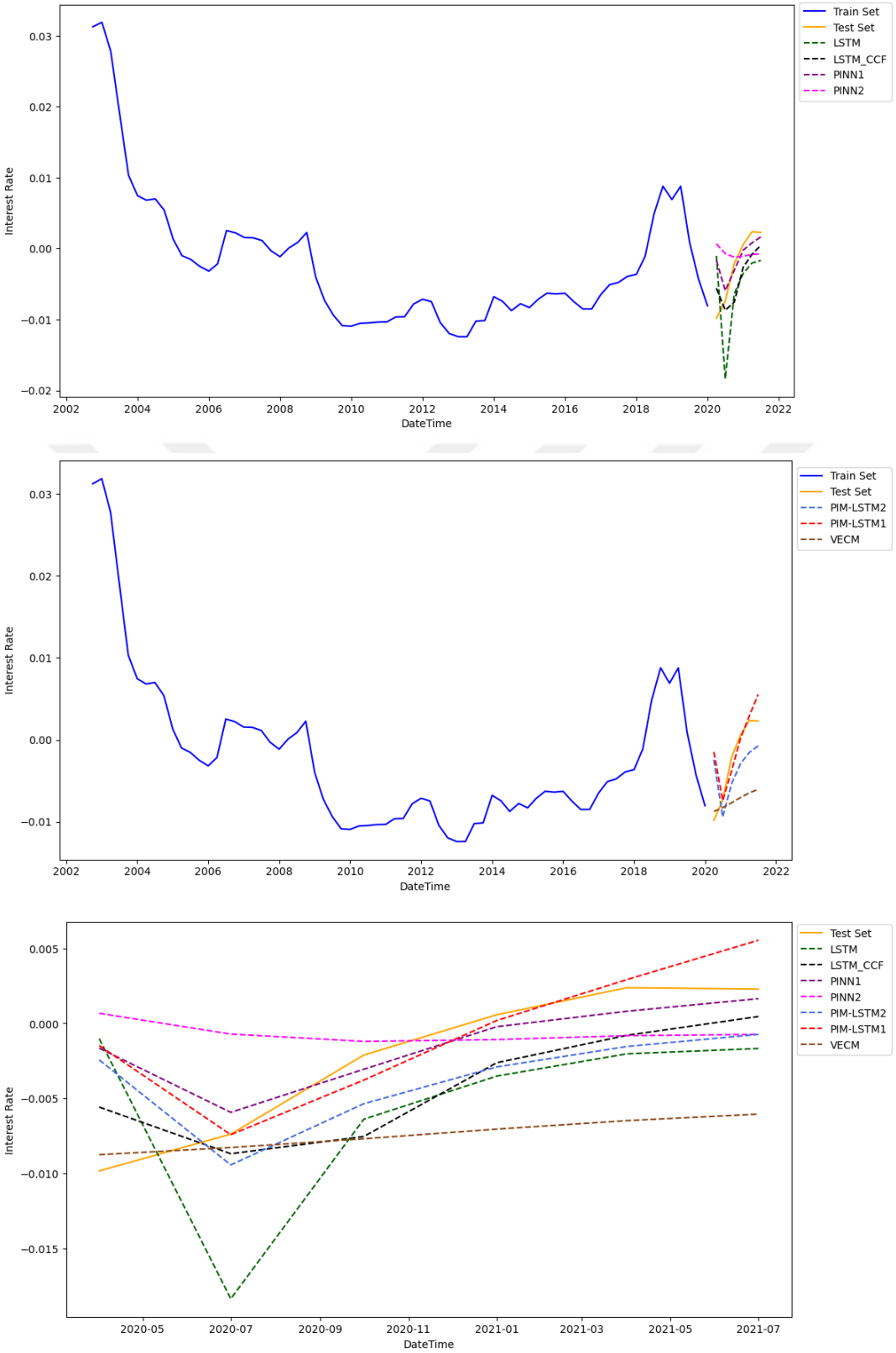

Figure 4.20 Forecast Plot of Turkiye's Interest Rate

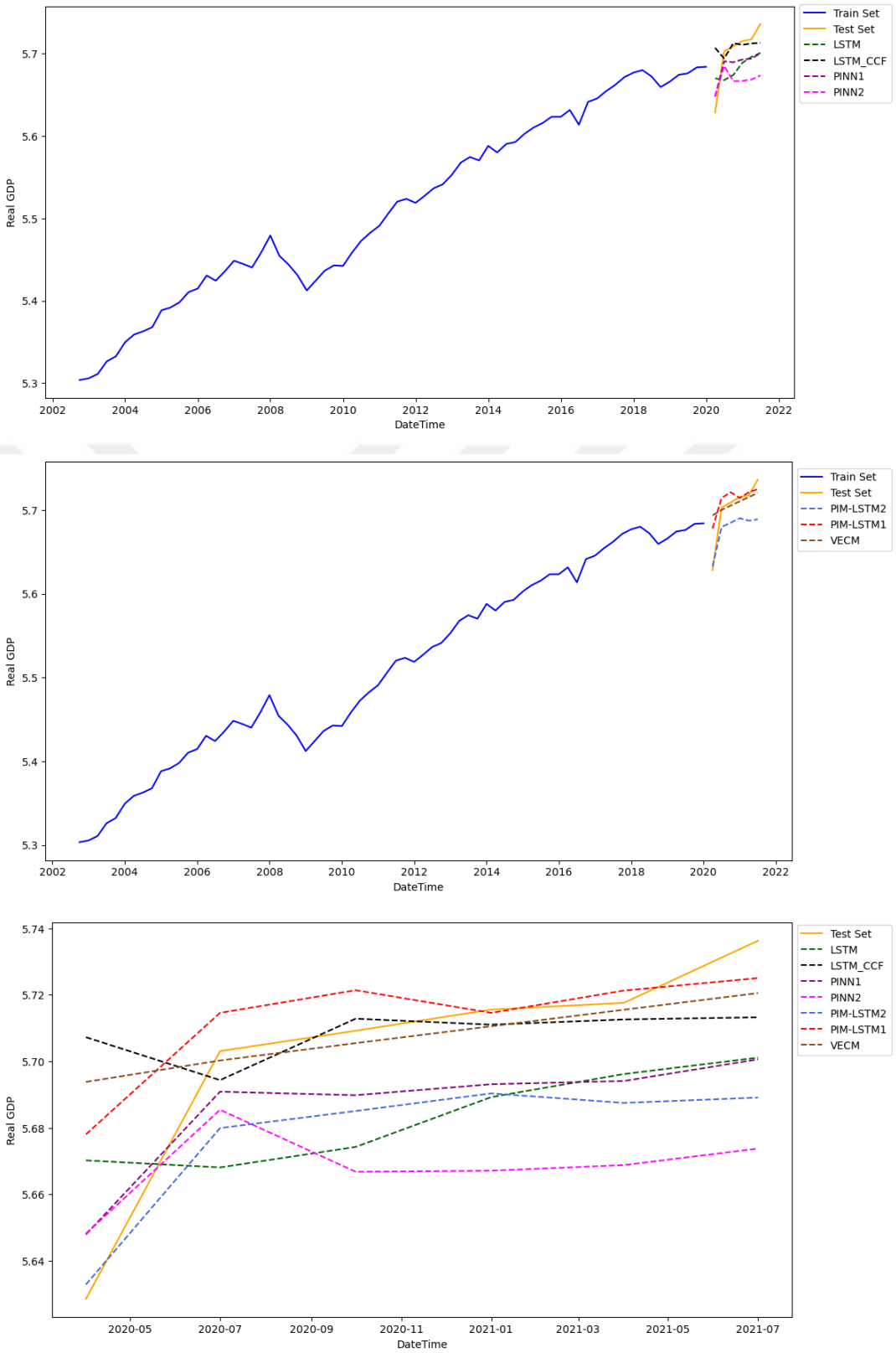

Figure 4.21 Forecast Plot of Turkiye's Real GDP

Secondly, we performed multivariate time series forecasting using the New Keynesian model for Mexico. We employed six deep learning models to forecast the values associated with the series in the New Keynesian framework. These models include the LSTM model, the LSTM model with CCF loss, the PINN model utilizing a Data-Driven Solution, the PINN model based on Data-Driven Discovery, the PIM-LSTM model with known parameters (referred to as PIM-LSTM1), and the PIM-LSTM model with unknown parameters (referred to as PIM-LSTM2). After performing hyperparameter tuning on all models for three series, the performance metrics, MAE, RMSE, and MASE, were calculated. Furthermore, the Vector Autoregressive (VAR) is implemented for multivariate time series forecasting. Table 4.30 shows the out-of-sample forecast performance of these models for each series. In terms of overall performance across all indicators, the PIM-LSTM2 achieves the lowest MAE (0.0030), RMSE (0.0036), and MASE (1.5847), indicating superior accuracy.

Table 4.30 Test Set Model Performance for Mexico's New Keynesian Model

Model	**MAE**	**RMSE**	**MASE**
LSTM Model	0.0130	0.0189	6.7887
PINN Model (Data-Driven Solution)	0.0087	0.0126	4.5628
PINN Model (Data-Driven Discovery)	0.0136	0.0191	7.1034
LSTM_CCF Model	0.0080	0.0103	4.2164
PIM-LSTM1	0.0033	0.0041	1.7471
PIM-LSTM2	**0.0030**	**0.0036**	**1.5847**
VAR Model	0.0064	0.0081	3.3501

Figures 4.22, 4.23 and 4.24 depict the time series forecast plot for the CPI inflation rate, the short-term interest rate and real GDP of Mexico. In these plots, the legend of "PIM-LSTM2" represents the PIM-LSTM model with unknown parameters, while "PIM-LSTM1" represents the PIM-LSTM model with known parameters.

Additionally, "PINN1" represents the PINN model with data-driven solutions and "PINN2" refers to the PINN model with data-driven discovery.

Figure 4.22 shows the performance of different models in forecasting the CPI inflation rate over time, compared to the actual CPI inflation values, which are represented by the "Test Set" in orange. Each colored and dashed line corresponds to a different model: LSTM, LSTM_CCF, PINN1, PINN2, PIM-LSTM2, PIM-LSTM1, and VAR. Although all models struggle to precisely capture the fluctuations observed in the inflation rate within the test set, the narrow range of the test set indicates that most of the forecasts are quite close to the actual values. Figure 4.23 shows the forecast of interest rates obtained by various models, including the LSTM model, LSTM_CCF model, PINN1, PINN2, PIM-LSTM2, PIM-LSTM1, and VAR. Overall, the PIM-LSTM1(red) model forecasts are closer to the actual interest rate values. Figure 4.24 exhibits the forecast of the real GDP using the LSTM model, LSTM_CCF model, PINN1, PINN2, PIM-LSTM2, PIM-LSTM1, and VAR. The PIM-LSTM1 (red) model provides a more consistent and realistic forecast than the other models.

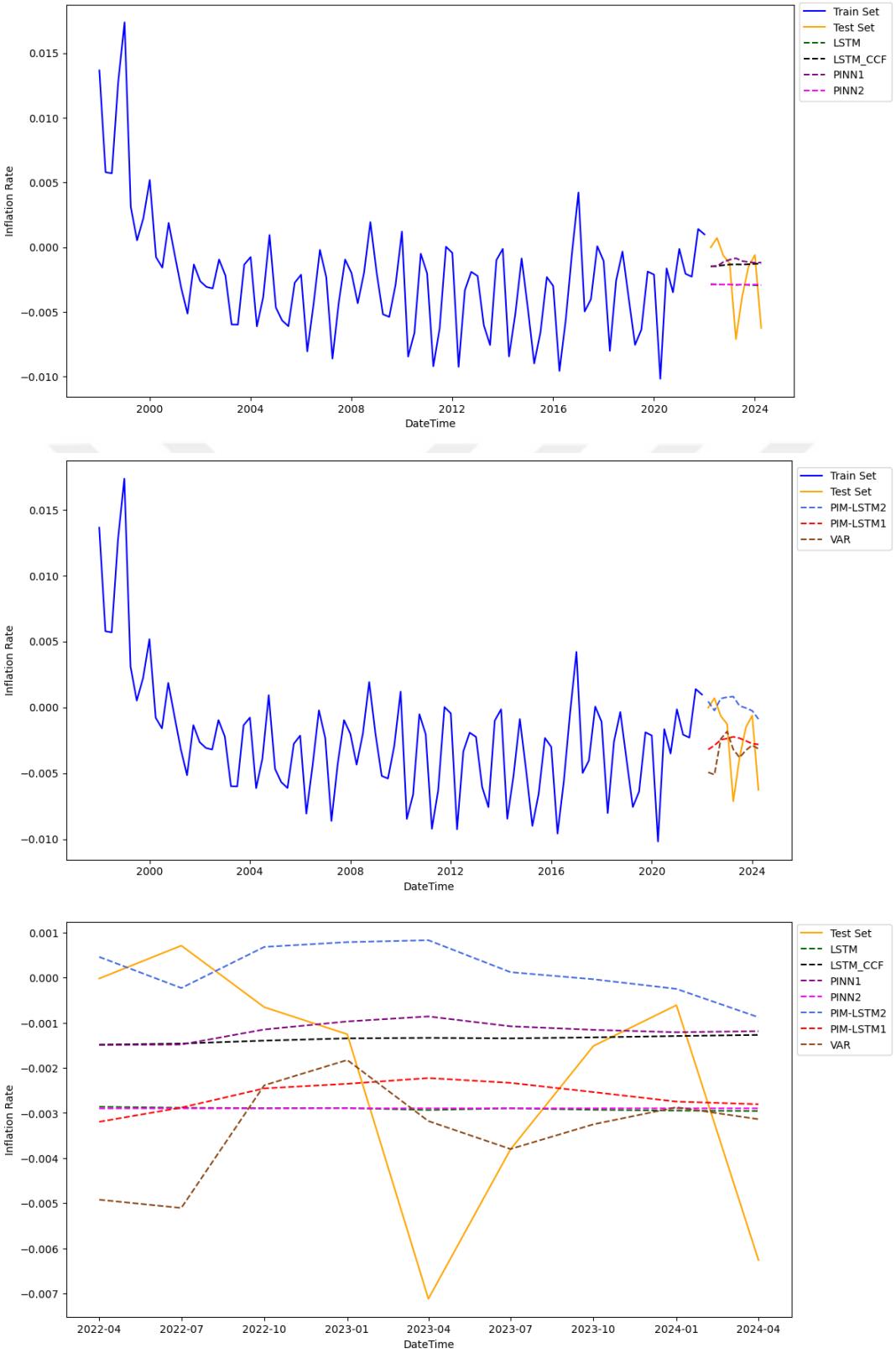

Figure 4.22 Forecast Plot of Mexico's Inflation Rate

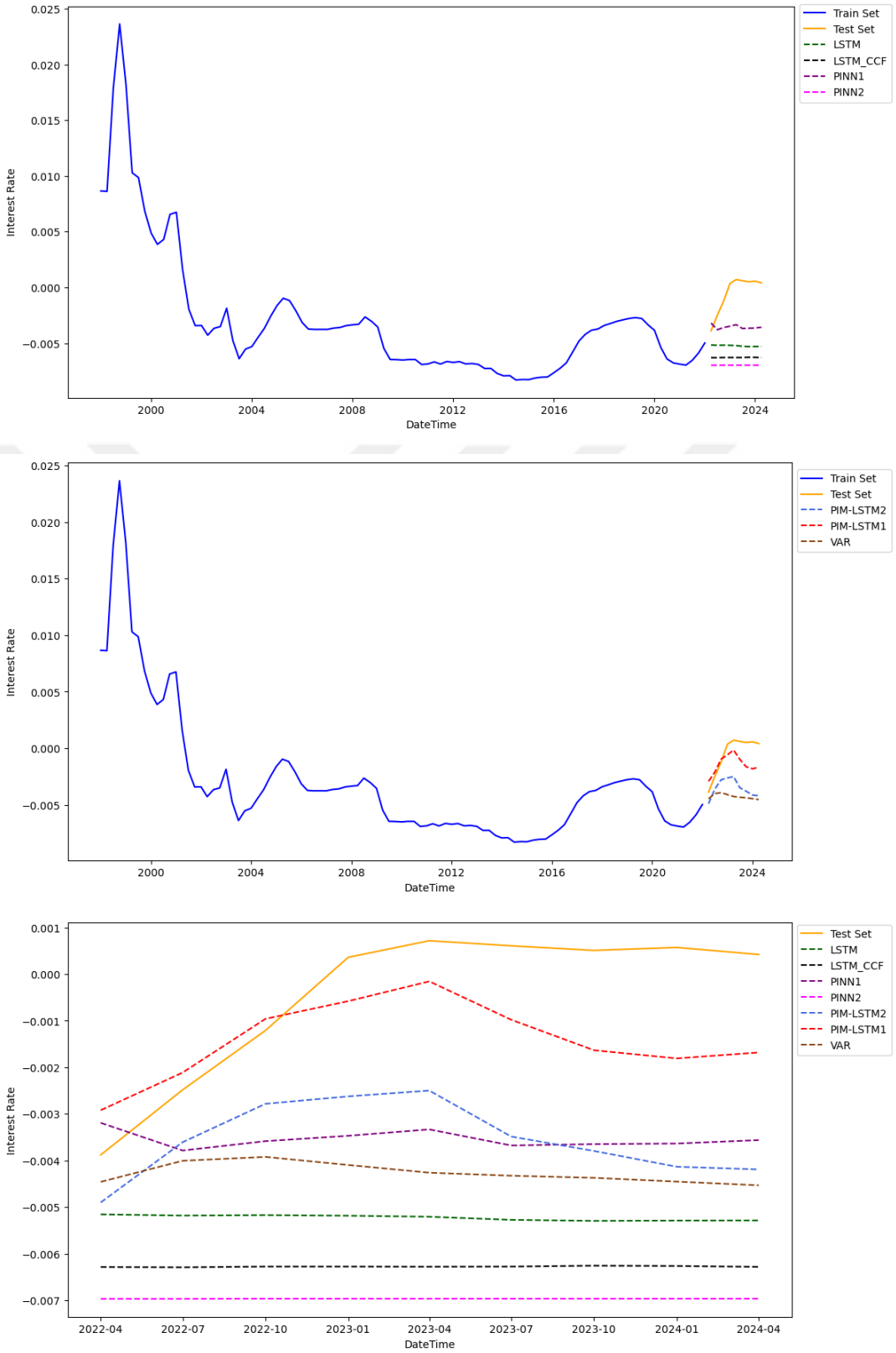

Figure 4.23 Forecast Plot of Mexico's Interest Rate

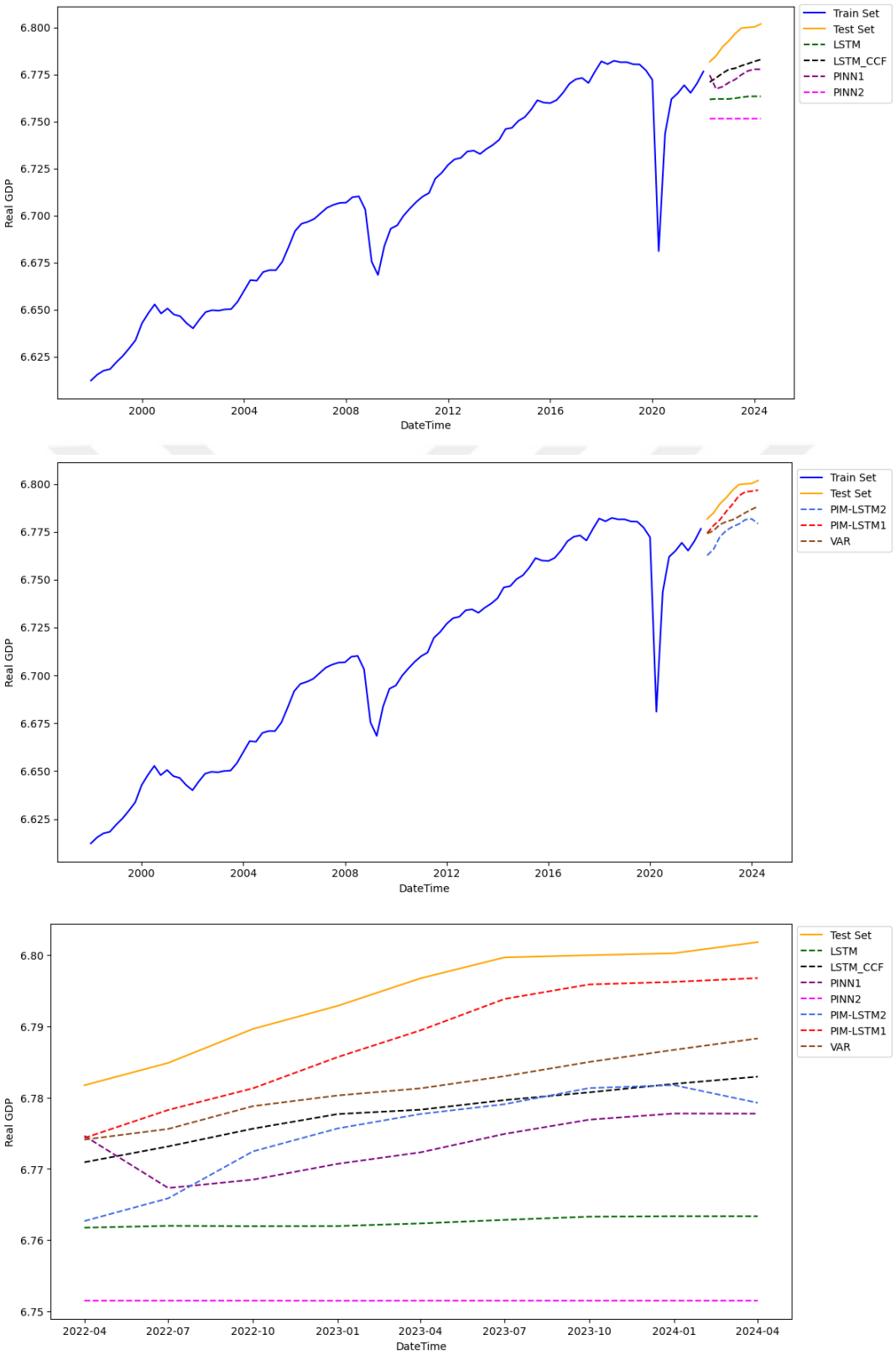

Figure 4.24 Forecast Plot of Mexico's Real GDP

Lastly, we conducted multivariate time series forecasting for the Dividend-Augmented Goodwin-Keen model (DAGKM) for the USA. Six deep learning models were utilized to get the forecasts of the series in the DAGKM model. These models are LSTM model and LSTM model with CCF loss, PINN model with Data Driven Solution, PINN model with Data Driven Discovery, PIM-LSTM model with known parameters (referred to as PIM-LSTM1) and PIM-LSTM model with unknown parameters (referred to as PIM-LSTM2). After performing hyperparameter tuning on all models for three series, the performance metrics such as MAE, RMSE and MASE for three series were calculated. Furthermore, VECM is built for multivariate time series forecasting. Table 4.31 presents the out-of-sample forecast performance of these models for each series. In terms of overall performance across all indicators, the PIM-LSTM1 achieves the lowest MAE (0.0080), RMSE (0.0103), and MASE (1.5546), indicating superior accuracy.

Table 4.31 Test Set Model Performance for US DAGKM model

Model	**MAE**	**RMSE**	**MASE**
LSTM Model	0.0190	0.0290	3.7129
PINN Model (Data Driven Solution)	0.0098	0.0174	1.9161
PINN Model (Data Driven Discovery)	0.0295	0.0475	5.7727
LSTM_CCF Model	0.0098	0.0149	1.9144
PIM-LSTM1	**0.0080**	**0.0103**	**1.5546**
PIM-LSTM2	0.0174	0.0265	3.3869
VECM Model	0.0399	0.0447	5.8167

Figures 4.25, 4.26 and 4.27 shows the time series forecast plot for the wage share, the debt ratio and the employment rate of the USA. In these plots, the legend of "PIM-LSTM2" represents the PIM-LSTM model with unknown parameters, while "PIM-LSTM1" represents the PIM-LSTM model with known parameters. Additionally, "PINN1" represents the PINN model with data-driven solutions and "PINN2" refers to the PINN model with data-driven discovery.

Figure 4.25 illustrates the performance of different models in forecasting the wage share over time, compared to the actual wage share values, which are represented by the "Test Set" in orange. Each colored and dashed line corresponds to a different model: LSTM, LSTM_CCF, PINN1, PINN2, PIM-LSTM2, PIM-LSTM1, and VECM. All models except the VECM (in brown) effectively showed an upward trend in wage share, as seen in Figure 4.25. Among these models, the PINN1 (in purple) and PIM-LSTM1 (in red) performed better at capturing the small fluctuations that followed the initial increase.

Figure 4.26 shows the forecast of debt ratio obtained by various models, including the LSTM model, LSTM_CCF model, PINN1, PINN2, PIM-LSTM2, PIM-LSTM1, and VAR. All models, except the VECM (brown), successfully captured the increasing trend in the debt ratio as seen in Figure 4.26. However, among these models, the PIM-LSTM1(red) model performs better, as its forecasted values are closer to the actual debt ratio values.

Furthermore, Figure 4.27 shows that all models, except the VECM (brown), successfully capture the increasing trend in the employment rate. However, the forecasts from the LSTM_CCF (black) and the PIM-LSTM2(light blue) are more closely aligned with the actual values.

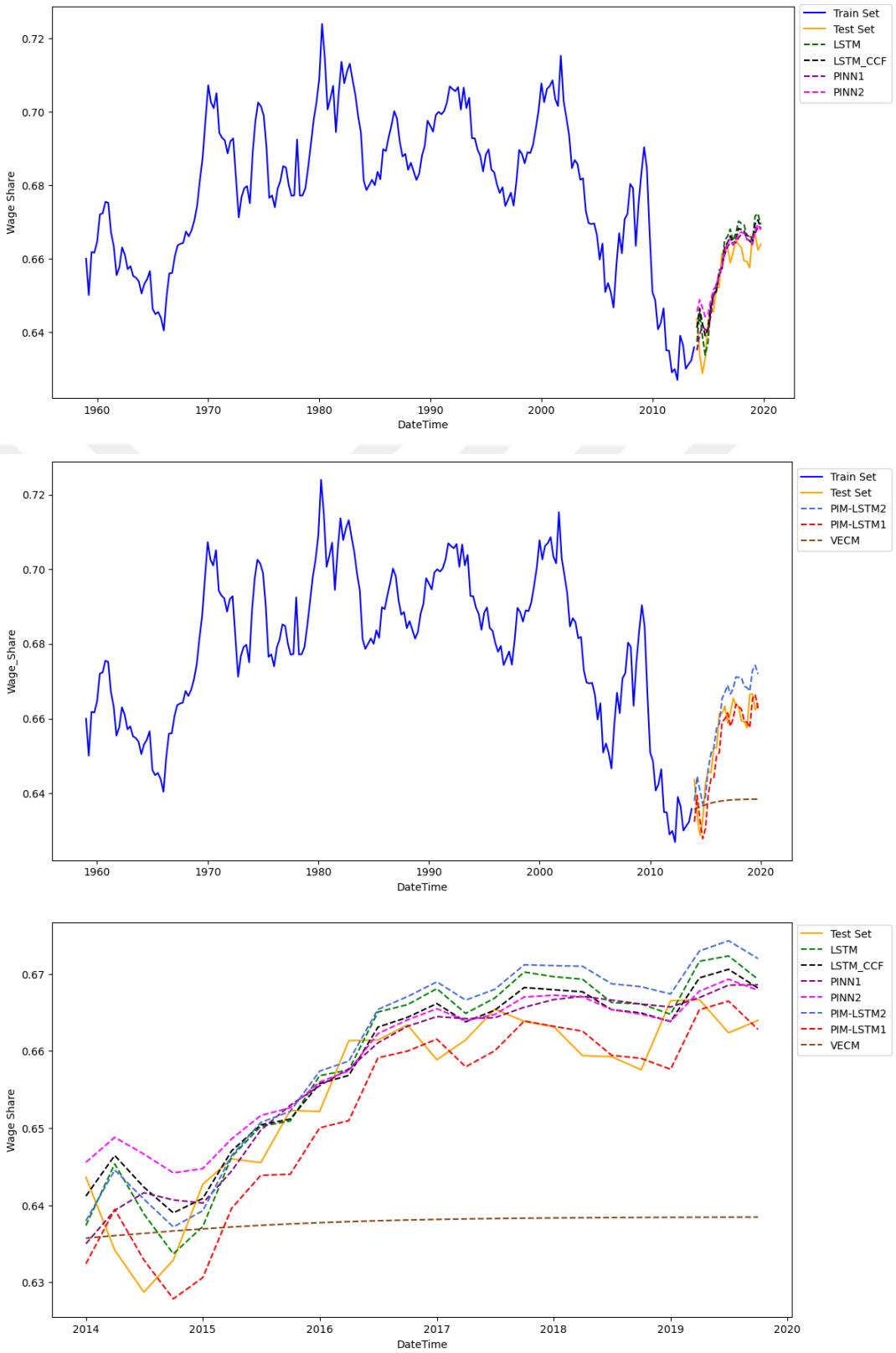

Figure 4.25 Forecast Plot of US Wage Share

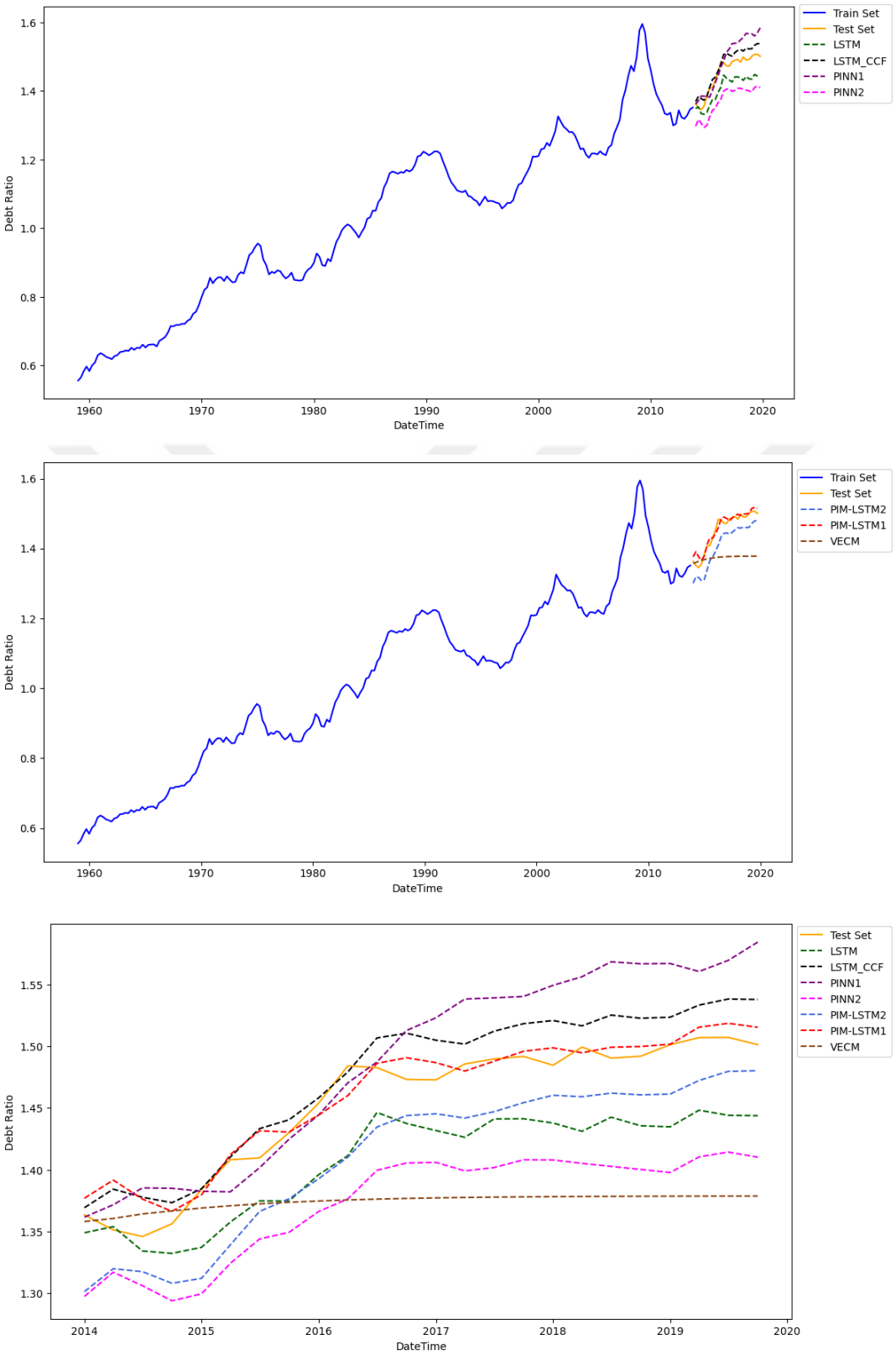

Figure 4.26 Forecast Plot of US Debt Ratio

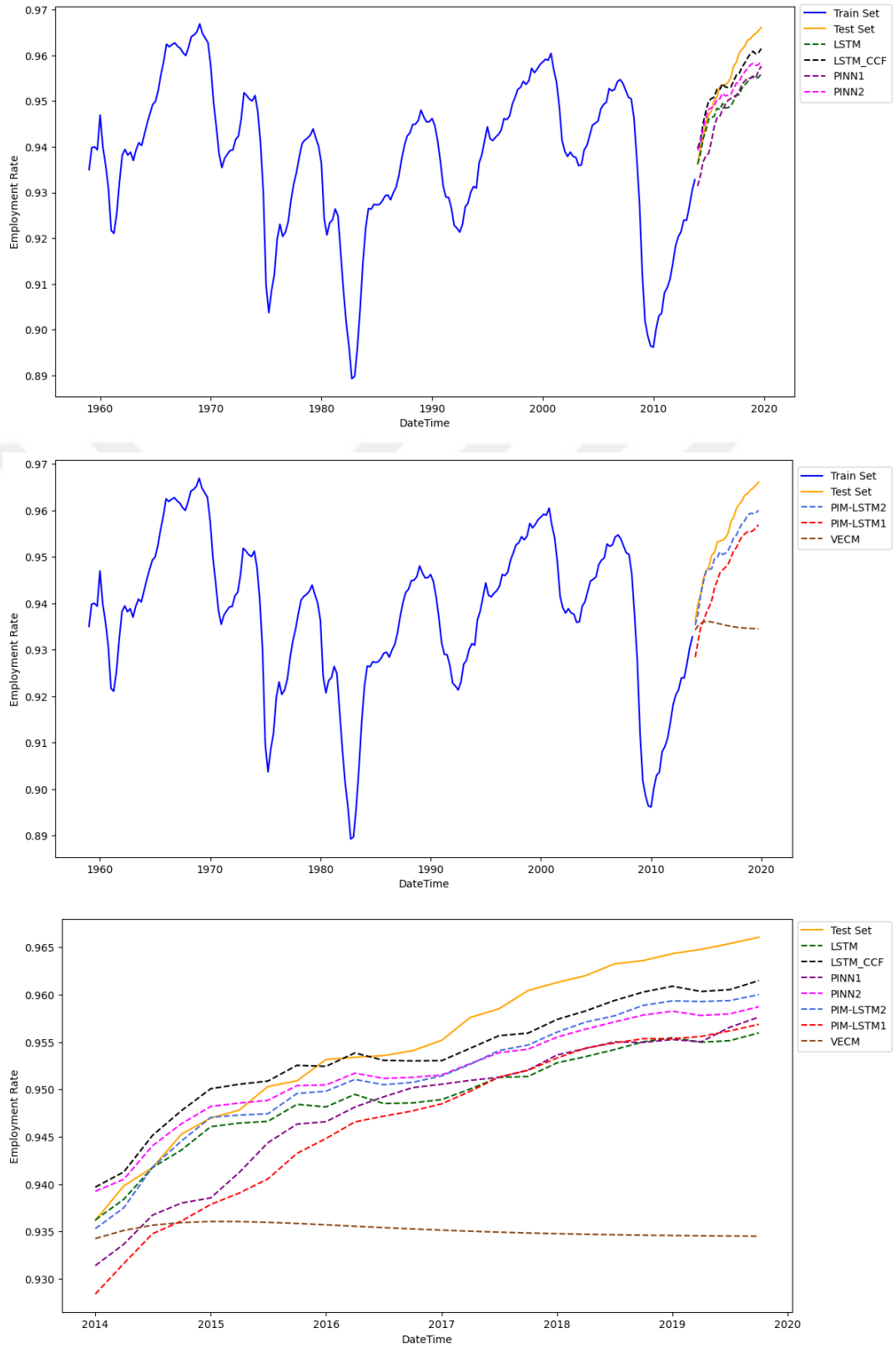

Figure 4.27 Forecast Plot of US Employment Rate

CHAPTER 5

DISCUSSION AND CONCLUSION

The main goal of this thesis is to adapt Physics-Informed Neural Networks (PINNs) to multivariate time series analysis, integrating causal relationships and cross-correlations into the PINNs framework to enhance model performance. Additionally, a side goal is to introduce the PINNs model to economic modeling, which has not been previously applied in economics. To achieve this, we developed the Prior-Informed Multivariate LSTM (PIM-LSTM) model, drawing inspiration from the PINNs framework. The proposed PIM-LSTM model leverages prior knowledge constraints and incorporates cross-correlation function (CCF) constraints to improve multivariate time-series forecasting. In this study, while training our model, the econometric model and the CCF function are incorporated into the loss function. This integration results in adjusting weights and biases according to the modified loss function, improving the training process and model performance. By integrating economic prior knowledge and cross-correlation information into the model, the PIM-LSTM model produces more reliable and consistent forecasts, ultimately improving the performance of multivariate time series forecasting. In this context, how the PIM-LSTM model works for the New Keynesian model for Turkiye and Mexico macroeconomic series is demonstrated, and its performance is compared with that of the LSTM model and the PINNs model. While training our model, the New Keynesian model and the CCF function are incorporated into the loss function to improve the overall training process and model performance.

In our analysis of Turkiye's macroeconomic series, the PIM-LSTM model with known parameters achieves the lowest errors across all performance metrics, demonstrating the highest overall accuracy for the New Keynesian Model. Specifically, the PIM-LSTM model with known parameters achieves a Mean Absolute Error (MAE) of 0.0073, a Root Mean Squared Error (RMSE) of 0.0132,

and a Mean Absolute Scaled Error (MASE) of 0.8013. In our analysis of Mexico's macroeconomic series, the PIM-LSTM model with unknown parameters achieves the lowest errors across all performance metrics, demonstrating the highest overall accuracy for the New Keynesian Model. Specifically, the PIM-LSTM model with unknown parameters achieves an MAE of 0.0030, an RMSE of 0.0036, and a MASE of 1.5847. Similarly, we show how the PIM-LSTM model works for the DAGKM model, and its performance is compared with the LSTM and PINNs models. Our analysis reveals that the PIM-LSTM model with known parameters achieves the lowest errors across all performance metrics, demonstrating the highest overall accuracy for the DAGKM model. Specifically, the PIM-LSTM model with known parameters achieves an MAE of 0.0080, RMSE of 0.0103, and MASE of 1.5546. Upon evaluating the performance metrics alongside the forecast plots in detail, it can be inferred that the PIM-LSTM model displayed the highest overall performance in multivariate time series analysis. Its ability to balance precision across different variables and effectively identify significant trends establishes it as the most dependable model for this analysis.

Future research could extend the PIM-LSTM model to additional economic models, such as DSGE or agent-based models, to evaluate its adaptability across different macroeconomic models. Alternative architectures, like graph neural networks or transformers, could also be explored to enhance forecasting accuracy. Improving interpretability through causal inference and explainability methods would help validate learned relationships against economic theories. Additional studies could examine the model's robustness under different economic conditions, including financial crises and changes in policy. These studies may use statistical methods like VAR or Bayesian estimation to create hybrid models. Improving training methods with new loss functions and better optimization strategies could also boost performance. Finally, extending PIM-LSTM to capture time-varying and nonlinear economic dynamics would help address structural breaks and evolving relationships in financial systems.

REFERENCES

[1] Arzani, A., Wang, J. X., & D'Souza, R. M. (2021). Uncovering near-wall blood flow from sparse data with physics-informed neural networks. Physics of Fluids, 33(7).

[2] Bai, Y., Chaolu, T., & Bilige, S. (2022). The application of improved physics-informed neural network (IPINN) method in finance. Nonlinear Dynamics, 107(4), 3655-3667.

[3] Bailly, H., Mortier, F., & Giraud, G. (2024). Empirical analysis of a debt-augmented Goodwin model for the United States. *Structural Change and Economic Dynamics*, *70*, 619-633.

[4] Bararnia, H., & Esmaeilpour, M. (2022). On the application of physics-informed neural networks (PINN) to solve boundary layer thermal-fluid problems. International Communications in Heat and Mass Transfer, 132, 105890.

[5] Box, G., & Jenkins, G. (1976). Time Series Analysis, Forecasting and Control. Holden Day.

[6] Brown, R. G. (1959). Statistical forecasting for inventory control. New York: McGraw-Hill.

[7] Cai, S., Mao, Z., Wang, Z., Yin, M., & Karniadakis, G. E. (2021). Physics-informed neural networks (PINNs) for fluid mechanics: A review. Acta Mechanica Sinica, 37(12), 1727-1738.

[8] Cai, S., Wang, Z., Wang, S., Perdikaris, P., & Karniadakis, G. E. (2021). Physics-informed neural networks for heat transfer problems. Journal of Heat Transfer, 143(6), 060801.

[9] Chiu, P. H., Wong, J. C., Ooi, C., Dao, M. H., & Ong, Y. S. (2022). CAN-PINN: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. Computer Methods in Applied Mechanics and Engineering, 395, 114909.

[10] Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition. J. off. Stat, 6(1), 3-73.

[11] Goodwin, R. M. (1967). A Growth Cycle: Socialism, Capitalism and Economic Growth, 1967, ED. CH Feinstein. In *Essays in economic dynamics* (pp. 165-170). London: Palgrave Macmillan UK.

[12] Güngör, M. S., & Güloğlu, B. (2019). The effects of structural shocks on macroeconomic fundamentals under aggressive monetary policy: The case of Turkiye. *International Journal of Business and Economic Sciences Applied Research (IJBESAR)*, *12*(2), 7-21.

[13] Hao, T. T., Yan, W. J., Chen, J. B., Sun, T. T., & Yuen, K. V. (2024). Multi-output multi-physics-informed neural network for learning dimension-reduced probability density evolution equation with unknown spatio-temporal-dependent coefficients. *Mechanical Systems and Signal Processing*, *220*, 111683.

[14] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780.

[15] Hu, Z., Jagtap, A. D., Karniadakis, G. E., & Kawaguchi, K. (2023). Augmented Physics-Informed Neural Networks (APINNs): A gating network-based soft domain decomposition methodology. Engineering Applications of Artificial Intelligence, 126, 107183.

[16] Huang, J., Wang, H., & Zhou, T. (2021). An augmented Lagrangian deep learning method for variational problems with essential boundary conditions. *arXiv preprint arXiv:2106.14348*.

[17] Jagtap, A. D., & Karniadakis, G. E. (2020). Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition-based deep learning framework for nonlinear partial differential equations. Communications in Computational Physics, 28(5).

[18] Jagtap, A. D., Kharazmi, E., & Karniadakis, G. E. (2020). Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. Computer Methods in Applied Mechanics and Engineering, 365, 113028.

[19] Jeong, Y., Jo, J., Lee, T., & Yoo, J. (2024). Combined analysis of thermofluids and electromagnetism using physics-informed neural networks. Engineering Applications of Artificial Intelligence, 133, 108216.

[20] Jiang, X., Wang, D., Fan, Q., Zhang, M., Lu, C., & Lau, A. P. T. (2022). Physics- informed neural network for nonlinear dynamics in fiber optics. Laser & Photonics Reviews, 16(9), 2100483.

[21] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. Nature Reviews Physics, 3(6), 422-440.

[22] Keen, S. (1995). Finance and economic breakdown: modeling Minsky's "financial instability hypothesis". *Journal of Post Keynesian Economics*, *17*(4), 607-635.

[23] Kharazmi, E., Zhang, Z., & Karniadakis, G. E. (2021). hp-VPINNs: Variational physics-informed neural networks with domain decomposition. Computer Methods in Applied Mechanics and Engineering, 374, 113547.

[24] Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2023). Neural operator: Learning maps between function

spaces with applications to pdes. *Journal of Machine Learning Research*, *24*(89), 1-97.

[25] Lawal, Z. K., Yassin, H., Lai, D. T. C., & Che Idris, A. (2022). Physics-informed neural network (PINN) evolution and beyond: A systematic literature review and bibliometric analysis. Big Data and Cognitive Computing, 6(4), 140.

[26] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

[27] Liu, L., Liu, S., Xie, H., Xiong, F., Yu, T., Xiao, M., & Yong, H. (2024). Discontinuity computing using physics-informed neural networks. Journal of Scientific Computing, 98(1), 22.

[28] Mao, Z., Jagtap, A. D., & Karniadakis, G. E. (2020). Physics-informed neural networks for high-speed flows. Computer Methods in Applied Mechanics and Engineering, 360, 112789.

[29] McClenny, L. D., & Braga-Neto, U. M. (2023). Self-adaptive physics-informed neural networks. Journal of Computational Physics, 474, 111722.

[30] Meng, Z., Qian, Q., Xu, M., Yu, B., Yıldız, A. R., & Mirjalili, S. (2023). PINN-FORM: a new physics-informed neural network for reliability analysis with partial differential equation. Computer Methods in Applied Mechanics and Engineering, 414, 116172.

[31] Nellikkath, R., & Chatzivasileiadis, S. (2022). Physics-informed neural networks for AC optimal power flow. Electric Power Systems Research, 212, 108412.

[32] Pang, G., Lu, L., & Karniadakis, G. E. (2019). fPINNs: Fractional physics-informed neural networks. SIAM Journal on Scientific Computing, 41(4), A2603-A2626.

[33] Pfeifer, J. (2018) *A Guide to Specifying Observation Equations for the Estimation of DSGE Models*, *University of Mannheim. Working Paper*. Available at: https://sites.google.com/site/pfeiferecon/dynare.

[34] Pourtakdoust, S. H., & Khodabakhsh, A. H. (2022). A deep learning approach for the solution of probability density evolution of stochastic systems. *Structural Safety*, *99*, 102256.

[35] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378, 686-707.

[36] Roberts, J. M. (1995). New Keynesian economics and the Phillips curve. Journal of Money, Credit and Banking, 27(4), 975-984.

[37] Roel, J. (2017). Understanding Recurrent Neural Networks: The preferred Neural Network for time series data. Retrieved from https://towardsdatascience.com/understanding-recurrent-neural-networks-the-prefered-neural-network-for-time-series-data-7d856c21b759.

[38] Rosenblatt, F. (1962). Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. Washington, DC: Spartan Books.

[39] Sims, C. A. (1980). Macroeconomics and reality. Econometrica: Journal of the Econometric Society, 1-48.

[40] Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, *375*, 1339-1364.

[41] Tanios, R. (2021). Physics Informed Neural Networks in Computational Finance: High Dimensional Forward & Inverse Option Pricing (Master's thesis, ETH Zurich).

[42] Taylor, J. B. (1993, December). Discretion versus policy rules in practice. In Carnegie-Rochester Conference Series on Public Policy (Vol. 39, pp. 195-214). North-Holland.

[43] Vadyala, S. R., & Betgeri, S. N. (2023). General implementation of quantum physics-informed neural networks. Array, 18, 100287.

[44] Wandel, N., Weinmann, M., Neidlin, M., & Klein, R. (2022). Spline-PINN: Approaching PDEs without data using fast, physics-informed Hermite-Spline CNNs. Proceedings of the AAAI Conference on Artificial Intelligence, 36(8), 8529-8538. https://doi.org/10.1609/aaai.v36i8.20830.

[45] Xu, Y., Zhang, H., Li, Y., Zhou, K., Liu, Q., & Kurths, J. (2020). Solving Fokker-Planck equation using deep learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *30*(1).

[46] Yang, L., Meng, X., & Karniadakis, G. E. (2021). B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. Journal of Computational Physics, 425, 109913.

[47] Yang, M., & Foster, J. T. (2022). Multi-output physics-informed neural networks for forward and inverse PDE problems with uncertainties. *Computer Methods in Applied Mechanics and Engineering*, *402*, 115041.

[48] Zhai, R., Yin, D., & Pang, G. (2023). A deep learning framework for solving forward and inverse problems of power-law fluids. *Physics of Fluids*, *35*(9).

[49] Zhang, X., Zhu, Y., Wang, J., Ju, L., Qian, Y., Ye, M., & Yang, J. (2022). GW-PINN: A deep learning algorithm for solving groundwater flow equations. Advances in Water Resources, 165, 104243

[50] Zendejas-Fonseca, A. S., Borrego-Salcido, C., & Venegas-Martínez, F. (2024). An Estimated DSGE Model Under the New Keynesian Framework for Mexico. *Computational Economics*, 1-24

APPENDIX A

TABLES FOR EXPERIMENT SETUP

Table A.1 Detailed data sources - quarterly and monthly data series

Variable	Sector	FRED code	Complete series name
Gross value added	Corporate	NCBGAVQ027S	Nonfinancial Corporate Business; Gross Value Added, Transactions
Gross value added	Noncorporate	NNBGAVQ027S	Nonfinancial Noncorporate Business; Gross Value Added, Transactions
Consumption of fixed capital	Corporate	BOGZ1FA106300003Q	Nonfinancial Corporate Business; Consumption of Fixed Capital, Structures, Equipment, and Intellectual Property Products, Including Equity REIT Residential Structures (NIPA Basis), Transactions
Consumption of fixed capital	Noncorporate	NNBCCFQ027S	Nonfinancial Noncorporate Business; Consumption of Fixed Capital, Structures, Equipment, and Intellectual Property Products, Current Cost Basis, Transactions
Net taxes on production and imports	Corporate	NCBPISQ027S	Nonfinancial Corporate Business; Taxes on Production and Imports Less Subsidies, Payable, Transactions
Net taxes on production and imports	Noncorporate	NNBTPIQ027S	Nonfinancial Noncorporate Business; Taxes on Production and Imports Less Subsidies, Payable, Transactions
Compensation of employees	Corporate	NCBCEPQ027S	Nonfinancial Corporate Business; Compensation of Employees Paid, Transactions
Compensation of employees	Noncorporate	NNBCEPQ027S	Nonfinancial Noncorporate Business; Compensation of Employees Paid, Transactions
Debt securities	Corporate	NCBDBIQ027S	Nonfinancial Corporate Business; Debt Securities; Liability, Level
Loans	Corporate	NCBLILQ027S	Nonfinancial Corporate Business; Loans Including Foreign Direct Investment Intercompany Debt; Liability, Level
Loans	Noncorporate	NNBTLBQ027S	Nonfinancial Noncorporate Business; Loans Including Foreign Direct Investment Intercompany Debt; Liability, Level
Time and saving deposits	Corporate	TSDABSNNCB	Nonfinancial Corporate Business; Total Time and Savings Deposits; Asset, Level
Time and saving deposits	Noncorporate	TSDABSNNB	Nonfinancial Noncorporate Business; Total Time and Savings Deposits; Asset, Level
Employment - non agri private sector	Non agri	LNU02032189	Employment Level - Nonagriculture, Private Industries Wage and Salary Workers
Unemployment - non agri private sector	Non agri	LNU03032229	Unemployment Level - Nonagriculture, Private Wage and Salary Workers
Employment - total private sector	Tot private sector	USPRIV	All Employees, Total Private
GDP deflator	All economy	GDPDEF	Gross Domestic Product: Implicit Price Deflator
Nonfinancial assets	Corporate	BOGZ1LM102010005Q	Nonfinancial Corporate Business; Nonfinancial Assets, Market Value Levels
Nonfinancial assets	Noncorporate	BOGZ1LM112010005Q	Nonfinancial Noncorporate Business; Nonfinancial Assets, Market Value Levels
Inventories	Corporate	BOGZ1LM105020015Q	Nonfinancial Corporate Business; Inventories Excluding IVA, Current Cost Basis, Market Value Levels
Inventories	Noncorporate	BOGZ1LM115020005Q	Nonfinancial Noncorporate Business; Inventories, Market Value Levels

APPENDIX B

ARCHITECTURE OF PIM-LSTM MODEL

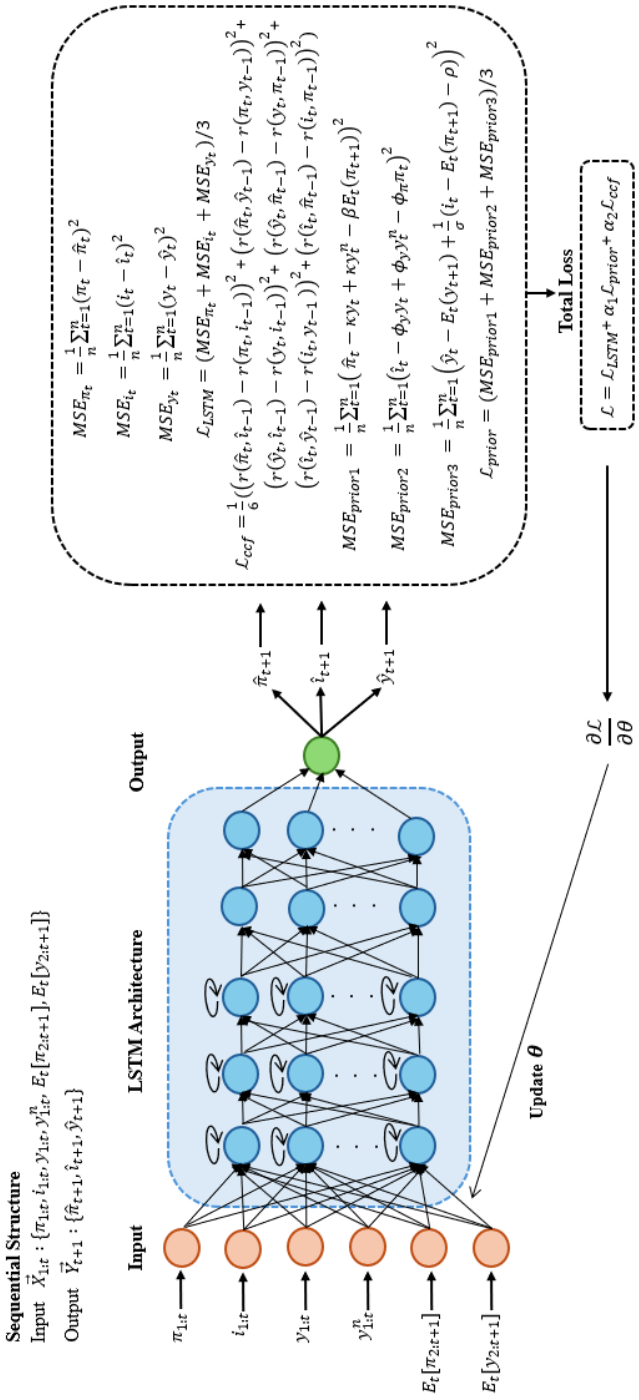

Figure B.1 Architecture of PIM-LSTM Model for New Keynesian Model

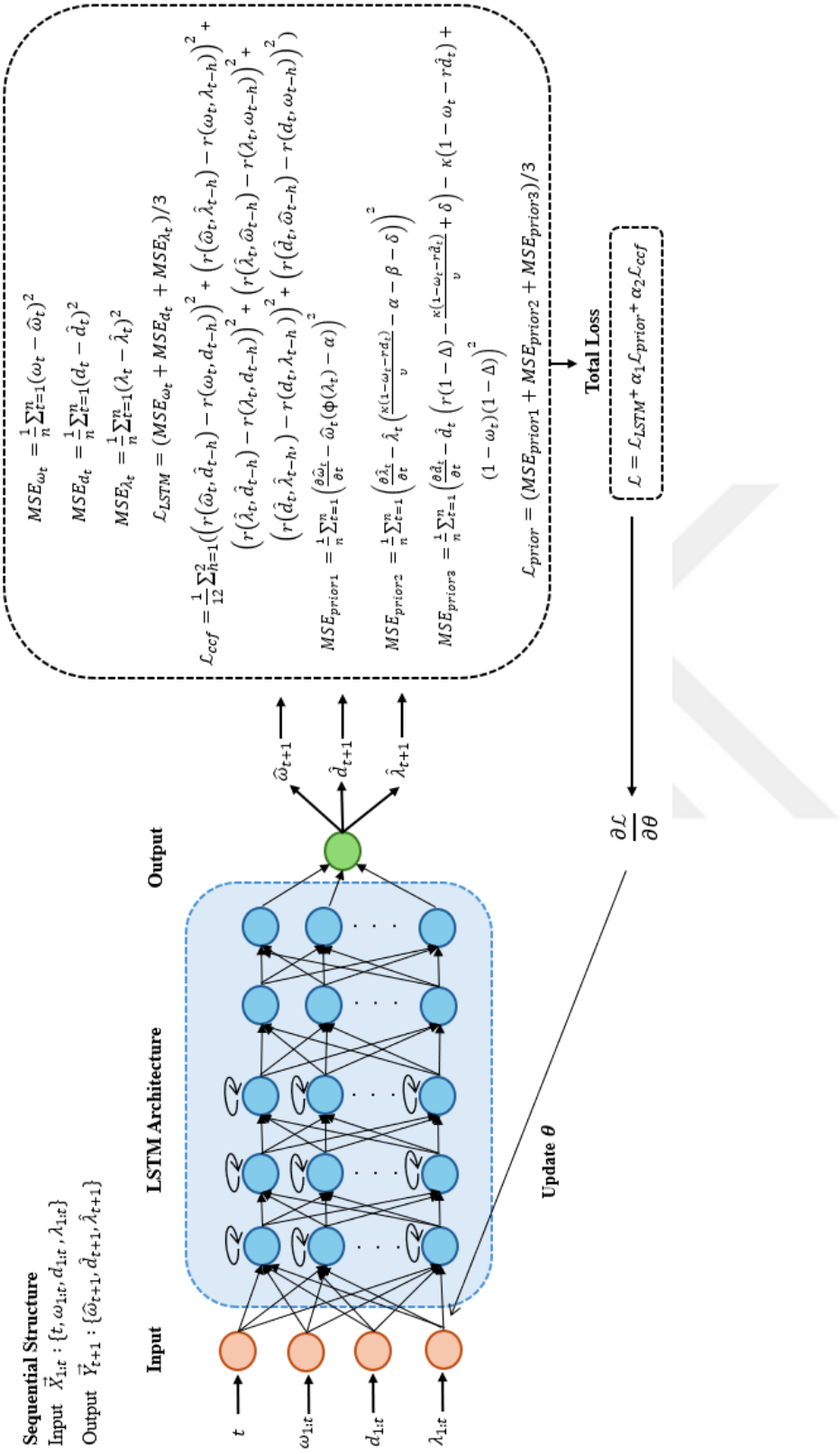

Figure B.2 Architecture of PIM-LSTM Model for DAGKM

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Aydemir Aydın, Petek

EDUCATION

Degree	Institution	Year of Graduation
MS	METU, Department of Statistics	2018
BS	METU, Department of Statistics	2015
High School	Atatürk Anadolu High School, Aydın	2009

FOREIGN LANGUAGES

Advanced English

PROFESSIONAL EXPERIENCE

Year	Institution	Position
2018-Present	METU, Department of Statistics	Research Assistant